Digital Mapping in Three Dimensional Space: Geometry, Features and Access

Aidan David Slingsby

Centre for Advanced Spatial Analysis & Department of Geography
University College London
University of London

A thesis submitted for the Degree of Doctorate of Philosophy at the
University of London
September 2006

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

I, Aidan Slingsby, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

Demand for large-scale digital vector base mapping is high, fuelled by developments in geographical information systems (GIS), spatial databases, and location-aware devices. The representation of real-world features (e.g. buildings, gardens, sheds), their properties and the spatial relationships between them are essential for supporting these types of applications and they provide an enduring focus for GIS research. There is an increasing and inevitable demand for three-dimensional (3D) data; however, many currently-available 3D data tend to focus on visualisation aspects, making them unsuitable for populating 3D feature-based databases for spatial analysis.

The thesis considers how 3D data can be structured in order that it may be used to support applications in a GIS context. The guiding design principles used to develop the conceptual model are:

- establishment of a data repository to which information can be added in an incremental fashion (in order that progress can be made without the requirement of exhaustive 3D surveys);
- storage of 3D geometry (facilitating the representation of complex multistorey and juxtaposed building parts);
- ability to describe different conceptualisations of features (e.g. 'rooms' and 'flats') and the relationships between them;
- seamless treatment of space exterior and interior to buildings (in order to treat all space with equivalence);
- incorporate pedestrian accessibility (spaces are topologically connected and pedestrian access constraints are embedded);
- representation of a temporal dimension.

The key concept is that of 'urban spaces' (discrete units of space in which human activity can occur) inside and outside buildings within the 3D environment. These are organised into layers whose surface geometries are interpolated, even where height data are poorly resolved.

The thesis develops a conceptual model, implements a prototype and then illustrates its use for various applications. Particular emphasis is placed upon applications which require pedestrian access information and the definition and identification of 'spaces' and 'real-world features' in 3D built environments.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my thanks to my supervisors Paul Longley and Harry Bruhns; Paul Longley for his high-level, strategic perspective on my work and how it fits in within the 'bigger picture' of GIS, for his support and for reading and providing feedback on my drafts so promptly; and Harry Bruhns for the early to mid evening exchange of ideas which substained me through my first year and his encouragement, particularly as the deadline for submission loomed. I would also like to thank Philip Steadman who, although not officially a supervisor, fulfilled the role of one admirably, and whose useful comments of my final draft were made whilst on his summer vacation. All have provided much-needed encouragement.

I have also been very fortunate to be part of the Centre for Advanced Spatial Analysis (CASA) and the postgraduate community there. The stimulating work environment, the culture of mutual support and friendship makes it an enjoyable place to work. This working environment is in large part stimulated by the vision of the director Mike Batty, to whom I would like to extend personal thanks, not least for offering me a desk in CASA during the critical final stages of my write-up, when my original desk was 'reallocated'. Both Paul and Mike were involved in obtaining the funded PhD studentship, without which I would not have carried out this project. Being part of CASA has given me many opportunities to meet many other researchers, providing a good perspective of a wide range of research, some of which have had direct relevance to my work.

I would like to acknowledge the help (particularly Chris Parker and Dave Capstick) and the financial support which I have received from Ordnance Survey, the industrial partner of the PhD studentship. The collaboration has provided me with a unique insight into Britain's national mapping agency.

Finally, I would like to thank friends and flatmates outside UCL for generally contributing to my well-being over the period and to my parents who support me in everything that I do.

Chapter 1

Introduction

National mapping agencies and similar organisations are major providers of geometrical geographical information; the details of such data provision organisations vary from country to country (Murray, 2002; van der Molen, 2003; Pluijmers, 2002; Bogaerts and Zevenbergen, 2002). Traditionally, the sole representation of these data was as paper maps. Within the past couple of decades, digital representations of such data have become increasingly important to support applications of spatial databases, geographical information systems (Lee, 1990; Morrison, 1994) and location-based services. This information will be referred to as 'base mapping' because it provides a context for a wide range of spatially-referenced data. First generation digital products were simply digital versions of paper maps (digitised 'drawings'); i.e. two dimensional and with a geometrical emphasis. There is increasing interest and research into the use of three-dimensional techniques for digital mapping. In addition, there is active research into other aspects of digital mapping apart from pure geometry (conceptualisations of real-world features, feature classification schemes, topological information including adjacency relationships, pedestrian access, vehicular access and the other inferences which can be made from the geometry, topology and attributes of the real world).

This thesis concerns large-scale vector topographic base mapping, at a sub-metre spatial resolution. The purpose is to abstract the real-world into a small and well-defined set of fundamental primitives for its digital representation. The focus is on urban areas, but the concepts can also apply to rural areas. This work is partly sponsored by the Ordnance Survey (GB)®[1], whose most up-to-date large-scale topographic vector base mapping product is OS MasterMap®[2]. Digital data provision is increasingly becoming the major share of sales from organisations such as Ordnance Survey. There is increasing interest in 3D mapping data and mapping organisations are investigating approaches to providing three-dimensional information and either

[1] Ordnance Survey® is a registered trademark of Ordnance Survey, the national mapping agency of Great Britain. (All occurrences of the name 'Ordnance Survey', refer to the Ordnance Survey of Great Britain.

[2] OS MasterMap®is a registered trademarks of Ordnance Survey, the national mapping agency of Great Britain.

providing or providing means of linking other non-geometrical information. This includes classified real-world features, relationships between features, postal addresses and access routes.

1.1 Three-dimensional geometry

Engineers and architects have well-established tools for creating, editing and maintaining three-dimensional (3D) geometrical models, which will be collectively referred to as computer aided design packages (CAD). These have traditionally been very geometry-centric tools. However, in the past couple of decades (Papamichael, 1997; Eastman, 1999; Khemlani, 2003a) domain-specific and semantically-rich 3D modellers and data exchange standards have been developed for buildings. They aim to support a wide range of different analysis tools for engineers and surveyors; e.g. visual impact, load, evacuation, fire, ventilation, lighting, amongst others. However, these models tend to be individual models modelled in isolation, rather than regional collections of building (van Oosterom *et al.*, 2006; Howard *et al.*, 1994).

Geographical information systems (GIS)[3] are designed for analysing parts of the environments on regional, national and global scales; in an urban context, the built environment. In contrast to CAD systems, they have poor 3D representation, handling and visualisation capabilities. In terms of representation, they are generally limited to surface models optimised for visualisation. In terms of spatial analysis – an essential component of GIS – they are largely restricted to 2D operations (an exception is line-of-sight modelling). The reason for the lack of 3D spatial analysis is that no commercial 3D GIS has the ability to resolve or represent full 3D topology[4] (Zlatanova, 2000); support for 2D topology is a defining characteristic of GIS. Reasons for the poor support for 3D are likely to include the balance between the redesign of the core software, the low level of mainstream consumer demand, the problems of assessing the geometrical validity of 3D data (snapping and slivers) and the computational overhead in resolving 3D topology. In general, mainstream consumer demand appears to be satisfied by the ability to do 3D visualisations, which can be implemented relatively easily on top of the existing algorithms.

In addition, there is little structured 3D spatially-referenced data available at the scale and coverage at which GIS operates; 3D LIDAR[5] GPS[6] points are unstructured – they represent clouds or linear sequences of points which usually need to be structured. A CAD file of a collection of buildings is usually too unstructured and at the wrong scale to be suitable for GIS analysis without substantial alteration

[3]This thesis will only consider vector GIS (rather than raster GIS).

[4]Topology geometry-independent spatial relationships.

[5]LIDAR ('Light Detection and Ranging') is a technology for measuring the distance to a surface using laser pulses. The result is a 3D point cloud from which the geometry of the surface can be interpolated.

[6]GPS ('Global Positioning System') is a technology for fixing a 3D position above the Earth's surface (the WGS84 ellipsoid) using information received from part of a constellation of orbiting satellites.

(though many GIS packages are able to display the 3D geometry). At the scale at which GIS normally operates, data are often abstracted to lower dimensions; e.g. buildings to polygons (footprints) or points (centroids of footprints). It is the *position of features* rather than the *precise geometry* which is the often the focus in GIS applications, in contrast to many CAD applications (Cowen, 1990; Newell and Sancha, 1990; van Oosterom *et al.*, 2006; ESRI, 2002). An example provided by Frank (1994a) is that even if 'London' and 'Scandinavia' have ambiguous boundaries, there are unambiguously outside each other.

Part of the reason for GIS' lack of focus on geometry compared to that of CAD software and geometry's abstraction to lower dimensional elements is the difference in application domains. CAD is designed for precisely representing 'as designed' structures for which the data are fully resolved. In GIS application domains, the data are often not fully-resolved.

There is increasing interest in applying the 3D concepts used in CAD to GIS. ESRI®[7] is the maker of the well-known GIS software ArcGIS® and AutoDesk® is the maker of the well-known CAD software AutoCAD®[8]. As stated, these are two separate families of software products. ArcCAD® is an engine for AutoCAD produced by ESRI, which adds some of the GIS capabilities from their GIS products (e.g. the building of topology and spatial analysis) into AutoCAD (ESRI, 2003). As the CAD and GIS markets have become more mature, this product is no longer marketed because each has implemented some of the others' functionality without the need for external modules. As GIS capabilities have become more ubiquitous, this trend can also be seen in mainstream databases, most of which now incorporate spatial data types and basic GIS type functionality (e.g. Oracle, MySQL and Postgres). Although 3D geometrical data types are supported in CAD packages, GIS packages (to a small extent) and many databases, 3D topology and spatial analysis are not supported.

As already mentioned, GIS and CAD applications tend to operate at different scales. GIS tends to deal with exterior space at city, regional and global scales, whereas CAD tends to deal with interior space at the building scale and potentially down to the scale of fittings and furniture. The scale at which CAD and GIS operates appear to be complementary; this is one of the main reasons why there is interest in integrating CAD and GIS file formats and concepts. For example, it can help support integrated planning by allowing the reuse of detailed 3D data of a specific building complex development to be used for simulations and planning at a variety of different scales including those traditionally handled by GIS. However, as discussed in Zlatanova and Prosperi (2006), there are many difficulties and unresolved issues associated with combining GIS and CAD data. Problems encountered through a series of case studies by (van Oosterom *et al.*, 2006) illustrate that as well as the

[7] ESRI®, ArcGIS® and ArcCAD® are registered trademark of Environmental Systems Research Institute, Inc.

[8] AutoCAD is a registered trademark of Autodesk, Inc.

practical software limitations of the exchanging data between CAD and GIS data formats, fundamental differences in data definition of scale and identity hamper this process. The case studies also show that data matching of GIS and CAD data in 3D is problematic because of the different primitives used, mismatches caused by different coordinate systems and scale, and different real-world feature definitions.

1.2 Real-world features

Both CAD and vector GIS software have their roots in pure geometrical modelling. However, their paths of evolution and development have diverged, with CAD focusing on precise 3D geometrical representation and GIS focusing on real-world feature representation and spatial analysis.

CAD packages traditionally described geometry as rather unstructured sets of geometrical elements representing a 'digital drawing', organised into themed layers. This is illustrated by the early *de facto* AutoCAD file format DXF which provides an insight in AutoCAD's representational ability at the time[9]. Only later were real-world concepts were incorporated. AutoDesk's Architectural Desktop (ADT) automates the geometrical modelling of real-world objects of interest to architects by essentially grouping together geometrical elements. This has been superseded by specialist parametric building modellers (Khemlani, 2002), which do allow real-world features to be described and classified according to feature catalogues and standards[10].

As seen in the previous section, GIS software has remained an essentially 2D product which often abstracts feature geometries to lower dimensions (e.g. polygons to points), concentrating on the position and extent of a real-world feature rather than its precise geometry. Real-world features are usually represented as points, lines or polygons whose non-geometrical aspects are described as attributes of features. Analyses can be performed on spatial and non-spatial aspects of features; for example, feature reclassification based on the spatial and non-spatial properties of a feature and its surrounding features. However, in general, the treatment of real-world features is no more advanced than stated here. Relationships between features (e.g. the nested set of 'room', 'storey', 'wing' and 'building'), fuzzy feature boundaries such as 'streams', 'rivers', 'estuaries' and 'lakes' (Burrough, 1994) and multiple conceptualisations of features are not handled except through geometry and relationship explicitly encoded in attribute tables. The various conceptualisations of real-world features tend to be application-dependent. In the last section, the observation of the importance of the structure of rooms, storeys and levels in the built environment was made. This corresponds to spaces within buildings. Since

[9] http://en.wikipedia.org/wiki/ASCII_Drawing_Interchange_file_format

[10] For example, the International Alliance for Interoperability's (IAI)Industry Foundation Classes (IFC, http://www.iai-international.org/Model/IFC(ifcXML)Specs.html) facilitate the exchange of semantically-rich building models; these are used for various aspects of the building construction projects (Eastman, 1999)

the boundaries of 'building' features and the boundary between inside and outside space are not always clear, a geometrical model which supports the representation of the insides of buildings will increase the scope for the ability to represent flexible feature conceptualisations.

In the GIS literature, there is much written about different aspects of real-world feature representation including definition, identity, classification, generalisation and temporal dimensions. Other than the provision of real-world feature classification catalogues (e.g. Ordnance Survey, 2001) and mappings, many of these issues remain unimplemented research questions. The temporal dimension of the real world is one of GIS' major weaknesses. No temporal analysis is offered (other than comparing changes between different snapshots of state) and recurring (low water and high water marks) or long term changes are not supported. As with the lack of support for 3D spatial analysis, the reasons that these are not supported are likely to be due to lack of customer demand, in turn partly due to the lower temporal resolution of most data, issues of temporal data validation and computational performance.

1.3 Access

The importance of topology for spatial analysis has already been explained. 3D topology describes non-geometrical spatial relations in three dimensions.

An important aspect of the environment, particularly the built environment, is that of access – specifically that of pedestrian access. Access as a topological relation between starting, intermediate and destination points is the basis for the access-graph (Steadman, 1983, p75), a network which describes topological connections between points (or spaces abstracted as points). The inspection of such graphs (e.g. figures 4.17 and 4.18) and the running of algorithms upon these graphs can yield interesting insights about how space is designed and used and also to perform tasks such as route-finding (e.g. through the use of Dijkstra's algorithm).

In this thesis, it is proposed that the integration of the topology of access within a feature-based geometrical model of the environment can be used in a number of useful ways. It can be used to validate (and perhaps contribute to assumptions about) the 3D geometry, because access to doors between spaces necessitates a geometrical relationship. It can also provide insights into how space can be used; as well as possible end uses, this might be a useful property for the definition of real-world features. Many applications require a description of pedestrian access, not just exterior transport networks, but within building and within the multilayered urban environment (Kwan and Lee, 2005; Meijers *et al.*, 2005).

1.4 3D base mapping

3D base mapping provide essential support for a wide range of uses (e.g. Lee, 2004b) and chapter 2 reviews such a range of GIS-type applications. Data models for representation and analysis are very application specific. The challenge for national mapping and similar organisations is to structure their data in such a way as to support the types of applications which rely on these data. This includes being able to retrieve 3D output of customised feature conceptualisations and customised access routes for particular applications.
Although comparisons have been made in this section between CAD and GIS, this thesis is not concerned with developing GIS or CAD technology, nor is it concerned integrating data between both systems. Instead, the discussion serves to illustrate and evaluate the different applications for which they tend to be used and the different technical approaches taken. What this thesis does focus on is *representation for storage* (rather than representation for analysis or visualisation) with a flexibility to output customised views for different customers and applications for spatial databases or GIS analysis.

This thesis proposes a pragmatic approach to structuring geometrical large scale vector data about the real world for *providing* 3D base mapping data for use with a variety of applications, in a GIS context. As the thesis title suggests, this task is divided into three interrelated strands:

- the representation of geometry
- the ability to represent flexible conceptualisations of real-world features
- the representation of pedestrian access

The representation of time in terms of all three of these strands will also be considered. The GIS context of this thesis positions the focus away from the CAD approaches of providing detailed 3D geometry and towards an emphasis on abstracting the geometry of the built environment such that it is appropriate for use by GIS for GIS-type tasks.

OS MasterMap and examples of its use, provide a starting point. OS MasterMap is a digital mapping product which is designed and maintained by Ordnance Survey. It abstracts real-world features into points, lines and polygons in 2D Euclidean space; so it is compatible with GIS. The real-world features (e.g. buildings, parks, road sections) are classified according to Ordnance Survey's published real-world feature catalogue (Ordnance Survey, 2001). Due to its use of a 2D coordinate system, the terrain height is ignored for real-world features, except for special spot-height point features. OS MasterMap also provides a fully-connected network for describing the connectivity of vehicular access, which can be integrated with the topographic mapping. This thesis proposes that such access topology should be an integral part of the mapping.

The points, lines and polygons in OS MasterMap are described in 2D space independently of the terrain. In a full 3D context, points, lines, polygons and polyhedra are described in 3D coordinate space and the elevation of the terrain becomes part of the geometrical description of the physical real-world features; though some abstract non-physical features such as land parcels and administrative boundaries can still be defined in 2D coordinate space.

The advantages of a full 3D model over the existing 2D model are that it is capable of representing the 3D geometrical shells of real-world features including roofs for buildings (providing height and more visual realism), multiple storeys or levels in the real-world and the interaction of the terrain with real world features (e.g. a building with entrances from the terrain on different storeys).

1.5 The problem

Full 3D geometry modelling environments (such as CAD packages) are designed to precisely represent the geometry of individual objects or sets of objects 'as designed' (rather than 'as built'), in isolation. For an environment for *storing* 3D data for use in a GIS, this thesis argues that the emphasis on a precise geometrical representation is not appropriate because it is very difficult to get fully-resolved up-to-date data to the required precision. This is especially the case at geographical scales of measurement if spaces within buildings are to be represented, because it is unlikely that data at the required detail will be available. Such data may become available at a later stage, in which case its addition to the (so far, incomplete) set should be facilitated. It is also unlikely that such detail is required for many GIS-type applications, so the absence of these data should not be a barrier to the production of mapping. In GIS, the focus for input data is not on precise geometry; rather, on the position and relative extents of particular identified features.

GIS software packages are the appropriate tools for the types of applications the framework designed as part of this thesis is designed to support. GIS software does not handle 3D data well so it is important to be able to collapse the data into 2D layers. In addition, 2D features (which tend to be non-physical administrative boundaries) should be able to be represented without being forced to be given an upper and lower extent.

Another problem is the definition of real-world features. In terms of ontology and semantics, there is an infinite range of possible end-use-dependent conceptualisations and labels of real-world features. For this reason, it is impossible to design a universal and exhaustive real-world feature catalogue suitable for all applications. In terms of geometry, the boundaries of different conceptualisations of features are dependent on the particular conceptualisation. Some features will easily nest within others and may form distinct hierarchies (e.g. 'room', 'flat', 'storey', 'building'). These are of interest here, because they describe the structure of the built environment. However,

many features will not nest; instead they will overlap in awkward ways (e.g. 'land parcel', 'building defined by main function', 'building defined by building material').

Access is usually described in the context of wider transport models (road traffic). Access does not usually extend to within buildings and is usually modelled separately from the geometry of the environment. Pedestrian access at the microscale which incorporates time- and pedestrian-dependent constraints is not usually applied at national mapping scale. However, it is an enormously important aspect of the built environment as described above. The embedding of a rich access model into the built environment would allow access constraints to be dependent on the geometry, topology and features of the environment.

1.6 Research questions and aim

This thesis examines options for representing and structuring data suitable for providing 3D national mapping data for use by spatial databases and GIS-type applications. The aim is to do this by abstracting the environment into a simple set of fundamental components which can be used as the building blocks for a variety of GIS-type applications. Higher level concepts will not be defined except where they are needed to support some of the core aims of the model[11]. However, a mechanism for defining new feature types with arbitrary geometrical extents will be provided.

1.7 The approach

As stated at the beginning of the previous section, this thesis proposes a pragmatic and novel approach for structuring geometrical large scale vector data about the real world in order to provide 3D base mapping for use with a variety of applications (chapter 2), in a GIS context. A result of a process of abstraction is the definition of a small set of fundamental concepts (section 5) which can describe the real-world to the stated requirements.

An essentially 2D approach is proposed in which 2D geometrical point, lines and polygons are topologically connected into distinct layers representing the ground terrain or floor. Absolute and relative heights are embedded within these layers in order to interpolate heights of the topologically-connected geometries within the layers. Pedestrian access topology and attributes are also embedded. Flexible real-world feature conceptualisations which can nest and overlap can describe their 3D through the use of the 2D footprint geometries and parameters describing their vertical form. Briefly, the motivations for this approach are:

- In terms of *data collection*, the existing high quality resource of large scale 2D mapping data (such as OS MasterMap) and existing spot height data can

[11] Features to support the access model were defined

be exploited without the need for a fully-resolved new 3D survey. Additional information can be added incrementally.

- In terms of *conceptualisation*, the concept of 'layers' is familiar in the real world when modelling phenomena are dominated by gravity; e.g. sediments, geological strata and most human activity[12]
- In terms of *compatibility with GIS*, the same geometrical primitives are used.
- In terms of *real-world feature representation*, multiple conceptualisations of features which are able to nest and overlap can be defined, whose 2D geometrical extent is described by a set of points, lines and polygons (organised into layers) and whose 3D extent (if appropriate) is parameterised.

This approach can be described as '2.5D', a term to refer to pseudo-3D models in which they are essentially 2D models in which the geometry is described in terms of the x and y axes, but where parameters are used to parameterise the geometries along the z axis. This is certainly not a new idea in itself, but the whole approach including the description of geometry, real-world features and access improves upon existing 2.5D approaches for the following reasons:

- Features are integrated with the terrain, rather than being placed 'on top' of a 2.5D surface terrain model.
- Each storey/level is treated with the same priority, rather than certain bridges, walkways and other raised platforms being considered secondary to the terrain.
- Vertical structures, such as steps and walls, can be represented.
- Incomplete height data can be incorporated.
- Data can be incrementally added, either to improve accuracy or to increase the spatial resolution.
- Space inside and outside buildings is treated seamlessly and 'buildings' can be subdivided horizontally as well as vertically.
- Multiple real-world feature conceptualisations can be defined and these can nest and overlap and change through time
- Detailed time- and pedestrian-dependent access topology and attributes are embedded.

These requirements are formally defined in chapter 3.

This approach to 3D modelling is clearly unorthodox when compared to the data intensive 3D modelling industry and, to a certain extent, when compared to national mapping which is based on precise measurements and capture specifications. However, this thesis proposes that this approach is both practical and useful, harnessing existing base mapping and adding well-defined constraints. More constraints can

[12]Most humans activities takes place in spaces on layers which form the lower boundary of these spaces.

be added over time as data become available, thereby increasing the quality of the interpolated model.

The realisation of the approach developed in this thesis considers features and spaces related to pedestrian access; for example, roofs are beyond its scope, but could later be added as a further feature type.

1.8 Development of conceptual and logical models and their evaluation

Chapter 5 describes well-defined concepts for realising the approach. These are then developed into a working implementation in chapter 6. These two chapters draw on the material from the earlier chapters including the theoretical and technical review in chapter 4.

Although chapter 6 describes a particular implementation, it includes detail which would be useful for building other implementations. The resulting prototype is an enormously useful tool in the development of the model and chapter 7 evaluates how effective the prototype and its underlying design is in fulfilling the aims of the approach. In general, it was considered to be a successful approach. Issues which require further development and ideas for further work are also presented.

The solution proposed by this thesis abstracts the real-world into a small and well-defined set of principles which are supported using a novel 2.5D system. The evaluation has shown that this may be viable for mapping organisations to structure their data and to gradually connect more height data. In the future, 3D concepts will become well-established. When this situation arises, the accumulated data will form the basis of a full 3D system.

1.9 Outline of the thesis

Part I contains two chapters which set the context in which the thesis is placed. Chapter 2 is an application-led review chapter which discusses the provision of large scale national mapping data and then provides case studies of application domains which rely on geographical data. Some of the solutions reviewed and the issues raised by this chapter are used to define a set of design principles, defined in chapter 3.

Part II concerns technical detail. Chapter 4 is a technical review chapter. Chapter 5 defines the implementation-independent concepts of the conceptual model. Chapter 6 then describes how the implementation was built in an object-oriented environment. Although this chapter concerns a particular implementation, the issues raised and examined are relevant for other implementations.

Part III reflects on the implications, successes and need for further work. Chapter 7 evaluates the model in the light of the design principles defined in chapter 3, which includes case studies of small examples tested with the prototype. This chapter also identifies problems with the model and discusses these. The thesis concludes with chapter 8, which includes suggestions for future refinement and future research.

Part I

Applications and Examples of User Requirements

Chapter 2

Spatial Data and Applications

This chapter establishes the context in which this thesis is placed. It first discusses the process of modelling of the real world[1], how this applies to base mapping and how digital base mapping has evolved (using Ordnance Survey products as case studies). This part of the chapter concludes with the assertion that there is a need for a feature-based model of the real world, capable of representing three-dimensional geometries and relationships. The application domain is then reviewed by comparing two families of tools (CAD and GIS), the approaches that they take and the types of applications they support. Finally, specific applications which require 3D feature-based information are reviewed and their needs are discussed. It is found that 3D geometry, the ability to identify multiple conceptualisations of features, pedestrian access and time, are all important aspects and that the incorporation of these into a digital feature-based large-scale national mapping database could support a wide range of applications which the data do not currently support. The material in this chapter is used to develop the requirements and design principles of the following chapter.

2.1 Models and frameworks

A complete representation of the entire real world is impossible. However, if a *subset* of the real world is taken which *only incorporates those aspects required for a specific application*, a complete representation is theoretically possible. The process of defining this formalised subset of the real world is the process of *abstraction*, a process which is more fully described in section 4.1. The realisation of an abstracted representation of the real world (or a fictitious world) is a *model*. A model may only be concerned with the *physical state* of the real world (i.e. an inventory) or it might be concerned with the *processes* which operate within it.

[1]The 'real world' is the collection of all facts, whether they are known or not (Kottman, 1999) and whether or not facts can be conceptualised.

Models can represent objects of the real world (as observed and measured) or of a fictitious world (pre-build construction plans, fictitious worlds in computer games and fictitious worlds in films). Models are necessarily simplifications. The way in which the real world is abstracted is important as it will affect the model's fitness for a particular use. A statue is a material model of a simplification of the *physical state* of a person or animal (usually the appearance). Henry Moore[2] is famous for his characteristic artistic interpretations of the human form as smooth, organic, curved sculptures whose lack of geometrical realism is in stark contrast to anatomical models of part of the body for informing about the structure of parts of the human body. Both types of model are different abstractions for different purposes. A hydrological model is an abstracted representation of the *process* of hydrology. Rather than representing a physical state, they represent (a subset of) the hydrological processes of a system. However, for their operation, they require a model of the state of the hydrological system.

In this chapter, it is models of *state* (inventories) which are being discussed.

The model itself may be a material object, drawing, diagram, photograph or digital representation. A statue is a material object which can be made of wood, clay, polystyrene or other material. A model of a building can be a material object built from similar materials, or it can be a photograph, a perspective drawing or a series of floor-plan diagrams. Material models of buildings can be used to present and demonstrate 3D information about real or planned buildings, and can be used in laboratories for modelling impacts on air flow and lighting, amongst others. Digital models, although not themselves material, can be the basis for the production of material models, through automating printing, etching and carving. Maps are models; they and geographical frameworks are discussed in section 2.3.

A *framework* structures the dataset it underpins, allowing the data to be interpreted. Non-digital models intended for human interpretation rely on existing human knowledge, which may be supplemented by other information, such as a legend. Paintings which are hung in art galleries may have a description explaining various relevant symbologies, analogies, conventions, the historical context, the social context and other information which may assist the viewer in interpreting the painting. Drawings and photographs normally use perspective, which is almost universally recognised. Engineering diagrams use well-known, sometimes domain-specific conventions in symbology and annotation. A map or set of floor-plan diagrams can usually be assumed to be universally understood.

The examples in the previous paragraph concern human interpretation which assumes existing background knowledge and the ability to interpret supporting information, such as legends. For *digital models* which require computer-based automatic interpretation, a more *structured and explicit* framework is required to represent all the knowledge required for interpretation.

[2] http://en.wikipedia.org/wiki/Henry_Moore

Without an explicit framework, the meaning of relatively unstructured data must be inferred. Scanned drawings, photographs and diagrams can be automatically interpreted in terms of visual pattern and colour. Optical character recognition (OCR) can resolve letters and characters from their clear and high-contrast images. Lewis (1996) and Lewis and Séquin (1998) describe a system for interpreting drawings of floorplans by just using the visual pattern of the scanned image. Techniques for the automatic and semi-automatic extraction of entities from aerial photography do exist, with varying success (section 2.2). Problems with such methods are noise in the images to be interpreted (due to the original paper copies being folded, smudged and the digital version being stretched or not having enough contrast) and that some of the rules required for interpretations are not simple, requiring some higher level of reasoning. A digital image library may only need to convert binary data to an on-screen representation (e.g. using the JPEG or GIF specifications). If the image library is needed to support the searching and cross-referencing of images, it would require a means to attach textural and numerical information to the images. The specifications of this would be part of the framework.

Digital models underpinned by detailed and explicit frameworks are more flexible than their material counterparts for analysis, because they can automatically and precisely interpret meaning. For example, digital building floorplans underpinned by a framework which allows semantic information to be embedded, improves the scope and success of automatic interpretation. Computer-based technology is increasingly being used to build computer-based models for buildings and geographical areas. Such digital representations are used as the basis for digital simulations such as air flow modelling, smoke and fire spreading, pedestrian evacuation and energy efficiency. Digital models and digital simulation and planning tools are becoming increasingly important in the building industry (Eastman, 1999); see section 2.6.4.

2.1.1 Geometry

The most important aspect of material models is their geometry, usually bearing a close resemblance to the real-world object. Technical diagrams usually also depict geometry, but due to the constraints of the 2D plane on which they are drawn or printed, 3D geometry is abstracted and is shown as 2D cross-, longitudinal or oblique sections or as perspective projections. Conventions such as 'cut away' diagrams or 'exploded drawings' are able to depict different aspects of an abstracted geometry. Non-technical drawings and paintings are likely to offer only a much looser interpretation and depiction of geometry.

Early digital models tended to concentrate on reproducing the geometry of technical drawings of 3D physical models; this is reflected in the history of the development of digital mapping (section 2.3.5) and computer aided design (section 2.6.1).

2.1.2 Non-geometrical aspects

Material models, drawings and diagrams often use colour to add richness to the model which may help in interpretation. A physical model of a building may colour code different sections to indicate ownership, activity or other similar attributes. Physical models and paintings may also use textural differences in the model surface to add richness. Diagrams use annotation, legends and subsidiary diagrams; since they are usually more technical, a stronger and more explicit framework needs to be in place in order to make interpretation as unambiguous as possible.

Early digital models tended to model geometry at the expense of non-geometrical aspects of the model. Models such as these provide only a weak framework and lack ontological and semantic richness; they were effectively only digital versions of drawings. This applied to early computer aided design software, whose drawings still had to be interpreted by a human. Advances in the computer modelling of processes were catalysts for these models to become ontologically and semantically richer, enabling them to be much more interoperable (Bittner *et al.*, 2006) and to be automatically interpreted and used in simulation models.

Geographic information systems have taken a different development path from computer aided design systems. In computer aided design software, the priority was to develop the means of storing precise geometries, curves and eventually full 3D geometry at the expense of the non-geometrical aspects of the models. In contrast, geographical information systems prioritised the linking of non-geometrical aspects of the data with the geometrical aspects. The geometrical aspects were not as well developed, apart from key areas such as support for projections and coordinate transformations. The fact that there are now many fewer technological constraints on the development of rich digital models has resulted in computer aided design systems obtaining more of the traditional functionality of geographical information systems and geographical information systems obtaining more of the traditional functionality of computer aided design. However, there are still major interoperability problems with combining data for both systems (Zlatanova and Prosperi, 2006). As well as the differences in scale and approach, the main problem is how to map the relevant (non-geometrical) concepts from one discipline to the other. It is now more of a theoretical rather than a technological challenge.

Digital models have broken the link in models between symbology and meaning. A wide range of geometrical and non-geometrical data can be stored and can be used to display data in a wide variety of customised ways. The challenge of designing frameworks and conceptual models is to capture as much richness as possible, using the richness to apply data in a wide variety of contexts and to be able to interoperate with other data and systems.

2.1.3 Time

The various states of objects in a state (inventory) model are valid for the particular instant in time at which the state of the object was last recorded. A model may contain states which were recorded at various times. The certainty of the currency of the information depicted in maps varies spatially.

As time passes, objects may change in state, gradually or instantaneously. As changes are detected and the states of objects are resurveyed, a corresponding change can be made in the attributes and geometries of the model's corresponding objects. Such changes can be managed by either completely replacing the old state with the newer state or keeping track of changes such that historical states can be reconstructed, compared and analysed.

Section 4.8 contains more detail about the representation of time and associated difficulties. An example of one such difficulty is deciding at what stage an object's change in state should result in its replacement by a new object with a new identity.

2.2 Data acquisition

A model (of state) is a framework populated with data; i.e. it cannot exist without data. There are various methods of collecting data, each of which results in data of varying qualities and richnesses, requiring different degrees of user intervention and being suitable for different frameworks. In the realm of engineering, building construction and computer gaming, it is usually prebuilt or fictitious objects which are being modelled; i.e. objects 'as designed' rather than 'as surveyed'.

2.2.1 Existing objects

For modelling existing objects, ground surveying techniques, aerial photography and satellite imagery are the main data sources. Tao (2006) provides a good review of three approaches for capturing the 3D geometry of objects: image-based, point-cloud based and hybrid.

Images can be interpreted and traced either manually or with the assistance of semi- or fully-automatic image processing techniques for identifying objects from an essentially unstructured set of pixel values. Boundaries can often be extracted and inferences about land use can be made (river, lake, field, park and road). Multiple overlapping imagery taken from different viewpoints can be the basis for the automatic reconstruction of 3D geometry (where the precise position, direction and camera specifications are known). Such imagery can be ground-based or taken from the air. Clearly, when using aerial photographs, objects and boundaries which have their view from the air obscured cannot be captured.

Scanners use various electromagnetic waves for detection and are capable of producing precise and dense unstructured point clouds. Airborne LIDAR systems can capture point clouds of sub-metre resolution[3] which can be interpolated to depict the surface of the Earth (including all the structures upon it; i.e. not the 'bare-Earth' model). An example of this can be seen in figure 2.5. Ground-based laser scanners are used for much more detailed 3D point cloud capture and have the advantage that they can be taken to spaces obscured from the air. Ground-penetrating RADAR[4] is being used in the Underworld project[5], a pilot scheme to map underground pipes and cables. As is the case for imagery, such data are rather unstructured and little more than raw geometry can be extracted, although characteristics of the returned pulses (e.g. scatter) can indicate the nature of the surface. A very active research area is the development of semi- or fully-automatic techniques for structuring LIDAR point clouds and aerial photographs (e.g. Grün, 1997), such as 'edge detection' to identify structures and the matching of subsets of points to templates of 3D objects.

Semi- or fully-automatic techniques are improved when the diversity of input data is increased, for example, using several wavelengths of light for scanners can improve the ability to infer land use.

2.2.2 Prebuilt or fictitious objects

Obtaining data for prebuilt or fictitious objects typically involves taking 'as designed' existing data from one framework and transforming it to be compatible with another framework. Frameworks, especially between different application areas, may be very different, with different scales, objects of interest, ontologies and semantics. The integration of large-scale data is comprehensively discussed in Zlatanova and Prosperi (2006). Case studies of integrating large-scale building data with smaller-scale data over large geographical are presented by Stoter and Ploeger (2003) and van Oosterom *et al.* (2006). They illustrate that this is not an automatic process and usually requires a significant amount of user intervention.

Ordnance Survey collects data on buildings for incorporation into their mapping through their CODES scheme[6]. This allows advance information about planned constructions to be incorporated into the mapping sooner than would otherwise be possible. Because of the problems with data interoperability, this is a manual process, but is manageable because the volume of submissions is relatively small.

In Queensland (Australia), the land registry sometimes registers land in three dimensions and routinely accepts existing three dimensional models for their incorporation into the digital land registry databases (van Oosterom *et al.*, 2006).

[3]Infoterra uses equipment capable of acquiring individually heighted data points at a rate of 33,000 per second, producing point densities of 1m and 1.7m and height accuracies of 15cm and 25cm from heights of 750m and 1450m, respectively. *Source: http://www.infoterra.co.uk/lidar_air.htm*

[4]RADAR ('Radio Detection and Ranging') is a similar technology to LIDAR, but uses longer-wavelength radio waves instead of light.

[5]http://www.comp.leeds.ac.uk/mtu/

[6]http://www.ordnancesurvey.co.uk/oswebsite/business/codes/index.html

2.3 Maps and geographical frameworks

Maps are models, usually of the ground surface. These are diagrammatic in style, in that objects are depicted as dots or their 2D projections from the ellipsoid used to approximate the local curve of the geoid-shaped earth onto a plane. Colours and labels are used to convey more information about them.

Early maps tended to be highly decorative illustrations, intending to be syntheses of the wider geography and culture of parts of the world. European interest in ocean navigation led to the founding of the French Académie des Sciences in 1666. This was partly responsible for a paradigm shift in mapping for navigation and planning, produced using scientific measurement and precise surveying equipment (Morrison, 1997). These resulted in paper-based, accurate, large-scale maps which could be used for navigation and planning.

Maps, however formal and scientific, are summaries of data. Numerical data (as measured) is summarised and hidden amongst spatially interpolated data (though the sample positions may be recorded; this is the case for spot heights). For example, terrain elevation data are usually sampled at specific positions (spot heights) or regular arrays of heights. Their treatment as a continuous surface necessarily requires interpolation. On maps, terrain elevation is often depicted using contours (isoheights: lines along which there is a fixed height). Both the form of the line (along its length) and its position (which affects the space) are likely to be the result of spatial interpolation from the original data points. The form and position of contours may also be affected by visual cartographic design issues. The necessary spatial interpolation leads to maps' inherent uncertainty which may be spatially variable (Funtowicz and Ravetz, 1990, ch 6). Any use of a map as a source of numerical data should be done with caution since the uncertainty in the resulting data is likely to be poorly understood. Maps (as any models) are abstractions; it is necessary to select which data are to be displayed to prevent overcrowding. In spite of this, prior to digital mapping, the map itself was usually the definitive source of the data, as the original tables of surveyors' measurements were usually discarded.

The omission of data on maps is a cartographic decision with reasons ranging from issues of design and clarity to more sinister reasons to do with prejudice, attempts to mislead and lie. Cartography is an art and maps are the products of human creation, embodying politics, prejudice and error (Funtowicz and Ravetz, 1990, ch 6).

Geographical frameworks are the means by which geographical information is distributed (Rhind, 1997), and maps which are produced through scientific measurement and surveying need to be based on graphical frameworks which, at the very least, define the projection, the scale and the datum. Typically for large scale maps, the framework provides a rectangular coordinate system which uses a map projection and a datum. In an increasingly digital era, where a higher degree of computer-

based interpretation is required, the geographical framework must be more explicit. Adherence of a set of spatial datasets to a high quality geographical framework allows them to be combined and related to each other. In general, national mapping frameworks are maintained by national mapping organisations. In a global context, the WGS84 coordinate system can be used, and it is possible to convert between this and the coordinate systems of national mapping frameworks, though this results in loss of precision.

2.3.1 National mapping organisations and base maps

National mapping agencies are major providers of base mapping and have produced definitive maps on regional or national scales for over two centuries. National mapping agencies vary widely in character and remit in different countries and have undergone a huge amount of change in the past couple of decades, fuelled by changes in technology, increased competition in the marketplace and changes in government funding priorities. As such, they operate in very different political, business and financial regimes (Murray, 2002; van der Molen, 2003). In Europe, a 'cost-recovery' model is generally favoured, where national mapping agencies fund themselves by selling data, and have a degree of financial freedom to respond to market needs. In the USA, the alternative 'open-access' model is favoured where data are made available for free and it is the private sector who takes the lead in research and development (Pluijmers, 2002). Some countries decentralise responsibility for maintaining regional data – this tends to be the case in countries with a federal style of government.

Base maps are models of the *state* of the world, providing geometry which is the basis for the 'mapping' of spatially-referenced data. Base maps may be *topographic* (section 2.3.2) where the geometry corresponds to physical detail (e.g. the boundaries of fields, roads or buildings) or *non-topographic* (section 2.3.3) where the geometry corresponds to non-physical detail (e.g. administrative regions). For this, base maps and the spatially referenced data must share a common framework. This framework may be a traditional coordinate system or it may provide a 'discrete georeferencing system' in which the base map geometry describes discrete positions or areas identified by unique identifiers through which non-geometrical data can be spatially referenced; e.g. postcodes and Census units.

Where the spatially-referenced data are geometrical or a georeferenced surface, base maps may be used to provide spatial context (e.g. GPS tracks superimposed on a topographic base map and population surfaces). Where the spatially referenced data are non-geometrical, data can be mapped through a discrete georeferencing system (e.g. statistics derived from the Census referenced to postcode area).

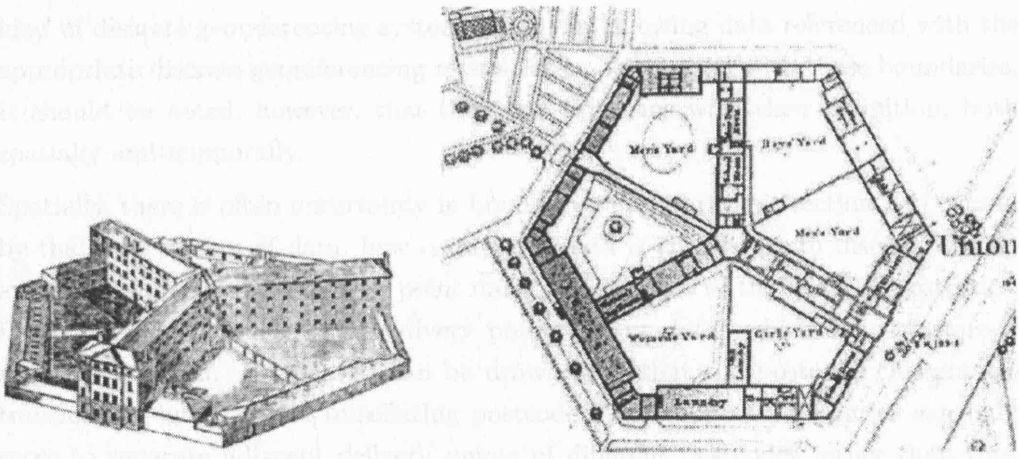


Figure 2.1: A 1:500 Ordnance Survey map of Abington Workhouse from 1875 showing ground floor detail. *Source: http://users.ox.ac.uk/~peter/workhouse/.*

2.3.2 Topographic base maps

Topographic base maps are models of the state of the physical detail of the world, produced by national mapping or similar organisations. National mapping organisations tend to map at a range of scales, but in many countries the role of mapping at the level of individual buildings is assigned to land registry organisations in order to support their land and building ownership databases (Bogaerts and Zevenbergen, 2002). In the Netherlands, the national mapping agency and land registry have merged (EC-INSPIRE, 2004). An alternative model for the production of large scale topographic mapping is a collaborative between different agencies to produce large-scale topographic mapping; this is the model used by the LVS GBKN Foundation in the Netherlands[7] which comprises the land registry, utility companies, telecommunication companies and the municipalities.

Large scale national mapping (depicting the shapes of individual buildings) does not show building interiors – this is considered to be at too fine a level of detail for national mapping agencies. However, 1:500 town plans published by Britain's national mapping agency for large cities published in the latter half of the nineteenth century did show the floorplans of some buildings (figure 2.1). With some exceptions, these were not published after 1900.

2.3.3 Non-topographic base maps

Non-topographic base mapping provides boundaries for non-physical units such as those used for administrative purposes, postcodes, Census output areas; more generally, socio-economic units (spatial units defined on the basis of social, cultural, economic or behavioural factors (Frank *et al.*, 2001, p9)). Where the boundaries of any of these can be mapped, they can be defined by a base map. Very often, some

[7] http://www.gbkn.nl/downloads/How%20to.ppt

kind of discrete georeferencing system is built in, allowing data referenced with the appropriate discrete georeferencing references to be mapped with these boundaries. It should be noted, however, that there are problems with their definition, both spatially and temporally.

Spatially, there is often uncertainty in boundary representation (section 2.4) caused by the fuzzy nature of data, how continuous data is classified into discrete classes and the definition of areas from point data. An example of the latter is postcodes. Postcodes refer to clusters of delivery points (point data) which form clusters if they are mapped. Boundaries can be drawn such that the postcode clusters are transformed into a set of tessellating postcode areas. These boundaries can only serve to separate adjacent delivery points of different postcodes; other than this, the position is rather arbitrary especially where there are no delivery points such as open spaces. Although the delineation of postcode areas is problematic and is not relevant to the original purposes for which postcodes were created (organising the postal system), they are important for mapping socio-economic data.

Temporally, the boundaries of socio-economic and administrative units are unstable over time (Frank, 2001). For example, the UK Census units between the 1991 and 2001 Census are different, requiring different versions of the Census unit base maps in order to map and analyse Census data from the respective years.

2.3.4 Paradigm shift in mapping

In the past few decades, the rapid development of computer hardware, software, computational methods and algorithms has changed most parts of our lives significantly. The impact of this on geographical information cannot be underestimated, because it has resulted in a vast amount of data which can be georeferenced in a range of useful ways. In many ways, this has driven the rise of geographical information science as a discipline and an industry, because new digital spatial representations and algorithms are needed to support the new industry.

It is a straightforward task to map any spatially referenced data in a geographical information system (GIS). Morrison (1994) describes a paradigm-shift in cartography (which he calls the "democratisation of cartography") where instead of the data provider employing a cartographer to render the information for consumption, the provider provides the raw data in a digital form. This gives the user more freedom to interpret the data and make cartographic decisions, though this is often constrained by some level of aggregation or summarisation of the data provided by the users. This applies to data that are aggregated to spatial units (this may be to protect privacy or just to make the data less useful) and any data which has been put into a classification scheme. Nevertheless, the user still has more control over the results than before. (This 'empowerment' of the user presents some problems, for example problems due to poorly described or interpreted metadata and the loss of the art of cartography – these concerns not discussed here).

This paradigm shift is important, because it separates the raw data from its display (this is by no means universally true however, because raw data which has been aggregated by area can only offer limited flexibility when it comes to mapping). Since raw data are usually superimposed on base maps depicting some of the geometry of the real world, another important effect is that there is more of a marked difference between the *topographic base map* and *'other mappable data'*. The 'base map' depicts the geometry that is necessary to put the other mappable information into context. In many cases, the 'other mappable data' requires, or would be enhanced by, their incorporation with base maps. Some of the 'other mappable data' may rely on the base mapping for its mapping (if discrete georeferencing is used) or may be mappable in its own right (if map coordinates are used).

Topographic maps (dependent on the scale and scope of the map) provide geometrical ground detail and are used as base maps. Large-scale topographic maps of cities depicting individual buildings and roads are good base maps for studying and mapping pedestrian movement patterns, building use and road congestion, to give a few examples. Smaller-scale mapping might be used to look at agricultural land use or soil types or nationwide economic information. Boundary base maps are essential for mapping data that are discretely georeferenced. Examples are the boundaries of Census collection areas and postcode areas – these are the most common ways of mapping socio-economic data.

2.3.5 Digital mapping

The paradigm shift in mapping should, in theory, reduce the problems associated with the inherent map uncertainty, because the map is no longer the definitive source of data; maps are instead a *view* generated from precise definitive data (in practice, the underlying data may be neither precise nor definitive). In terrain modelling, the *original surveyed points* can always be used to *generate different surface interpolations* where required. However in practice, some of the 'definitive' set of data may be still sourced from an imprecise or interpolated source.

Within the past decade, many mapping agencies have undertaken digitisation programmes and now definitively store their maps in databases, from which they are managed, edited and output as either paper maps or digital mapping products. Since these are sourced from paper maps, it is likely that inaccuracies and a loss of precision from the originally-surveyed data exist. However, as the data are maintained and updated, such inaccuracies are gradually removed.

In common with most programmes involving the wholesale digitisation of information, initial efforts have been little more than digital versions of the original paper versions. Ordnance Survey observed that customers were increasingly structuring the data themselves for use in a GIS (Barr, 2004) and embarked on a programme of structuring their data, in which the linework has been structured into polygons representing 2D spatial extents of conceptualisations of real-world features.

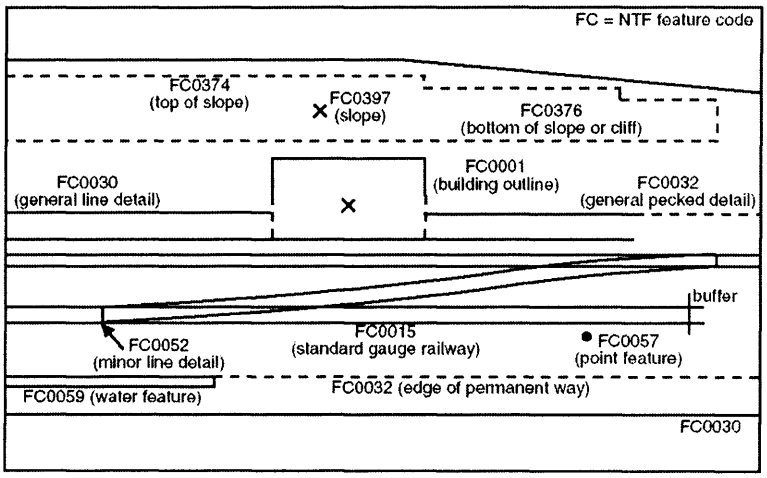


Figure 2.2: Example (showing part of a railway) from the Land-Line® User Guide. It shows points and lines classified with a 'feature class' (preceded by 'FC'). *Source: Ordnance Survey (2002, figure 4.8, p4.18). Ordnance Survey © Crown copyright. All rights reserved.*

This perhaps represents a transition to a *new* paradigm in cartography – the change from a relatively unstructured database of geometries representing map-based linework to a real-world object-centred database. This transition will be illustrated with a case study of Ordnance Survey's large-scale digital maps. Ordnance Survey completed its nation-wide large-scale digitisation programme in 1995.

2.3.5.1 Geometry-based digital mapping (case study: Land-Line)

The result of the completion of the wholesale digitisation of Ordnance Survey's paper maps was the creation of the National Topographic Database (NTD) in 1995, on which the digital data product Land-Line® is based (figure 2.2). NTD is very map-centric framework, in that an unstructured set of linework has been digitised. Amongst the first types of software for dealing with line-based digital vector data were computer aided design packages (CAD; section 2.6.1) widely used in engineering and the construction industry (section 2.6.4). The linework corresponds to the boundaries of areas (these are implicit) and linear features (Ordnance Survey, 2002). Points (dots) corresponding to particular positions of interest (e.g. spot heights, telephone boxes, wells and milestones) complement the linework. Each of these points and lines is termed a 'feature' and is allocated a 'feature class' which can be seen as corresponding to a type of linework or symbology (e.g. a building outline, base of a slope or cliff or a point feature). Land-Line does not have areal features. This is a major limitation because many geographical features are conceptualised by their areal extents rather than their boundaries; examples being fields, building footprints and land parcels.

Since Land-Line features refer to line-work on a paper map rather than conceptualisations of 'real world' features, Land-Line is most appropriate for the production of customised maps for printing and display. Where areal features are required

Figure 2.3: The polygon features of the Topographic Layer of OS MasterMap®. Each polygon has a unique identifier ('TOID®') which tends to correspond to a 'real world' areal feature such as a building or segment of road. In this figure, polygons are coloured according to the category of 'real world' feature. *Source: OS MasterMap® data displayed in ArcGIS® 8. Ordnance Survey © Crown copyright. All rights reserved.*

(such as individual building footprints for a GIS analysis task), polygons have to be built from bounding 'building outline' lines, a task achievable in many geographical information systems (GIS).

The geographical framework on which Land-Line is based is the same as that used for the original paper map; i.e. the National Grid (Britain's coordinate-based georeferencing system) and the set of specifications defined in the user's guide (Ordnance Survey, 2002) which includes detail on the types of topographic detail provided and how they are classified.

2.3.5.2 Towards feature-based digital mapping (case study: OS MasterMap)

The programme of structuring of the Land-Line 'spaghetti' into the feature-based OS MasterMap® was launched in 2001. Two of OS MasterMap's important characteristics are:

- Area (polygon) features exist in addition to the line and point features present in Land-Line. Polygons are exhaustively tiled (they do not overlap and there are no gaps), so road and pavement segments are represented as polygons too. Since many geographical features are often conceptualised as land parcels or footprints on maps, these new polygon features more closely represent these addressable objects which are conceptualisations of real-world features.
- Since OS MasterMap features have a closer match to addressable objects in the real world, they are each allocated a unique identifier called a 'TOID®' ('topographic identifier'). A TOID uniquely references an OS MasterMap feature and its corresponding real-world feature (if applicable). A small change in a real-world feature may be small enough to allow it to retain its TOID (fig-

ure 4.19); a substantial change in a real-world feature may result in it being allocated a new TOID. However, OS MasterMap features do not necessarily correspond to real-world features, particularly those which are lines and points; so a TOID may not correspond to a real-world feature at all.

OS MasterMap is split into several themes called 'layers'. The point, line and polygon features are contained in the 'Topographic Layer'. Road accessibility and connectivity information is provided as a fully-connected 2D geometrical network of roads in another layer called the "Integrated Transport Network™(ITN)". Nodes of the network represent the positions of junctions and link the topologically-connected lines. Lines represent the 2D geometry of road centrelines and lines can cross without being connected (this is the case when they are on different levels). Nodes and lines of the network have associated traffic access and restriction information. However, network connectivity cannot be derived from the road polygons in the Topographic Layer because the 2D depiction of geometry results in over-passes cutting the connectivity roads of underpasses. The network is also annotated with constraints upon access to the network as attributes (more detail is provided in section 2.9.3).

2.3.5.3 Frameworks to support feature-based mapping (case study: Digital National Framework)

The framework which OS MasterMap adheres to is much more explicit than that upon which Land-Line is based. The Digital National Framework™(DNF®) is based on the same coordinate-based georeferencing systems, but it has its own set of principles and its own discrete georeferencing system (through the use of the 'TOID').

DNF is presented at http://www.dnf.org/ as an open set of mapping principles enabling data in Britain to be more easily shared. It is defined in a white paper (Ordnance Survey, 2004), along with a rather free discussion of possible developments for the future. Rather than attempting to be a formal standard, it aims to be a set of best practice guidelines to facilitate data sharing.

DNF is defined as "... a model for the integration of geographic information of all kinds – from national reference datasets to application information at the local level" (Ordnance Survey, 2004, p13). It has the following guiding principles(Ordnance Survey, 2004, p13):

- The concept and methods shall be driven by the strategic needs of the wider GI community and the needs of the information industry.
- Data should be collected only once and then re-used.
- Reference information/data should be captured at the highest resolution whenever economically possible.
- Such information may then, where appropriate, subsequently be used to meet analysis and multi-resolution publishing requirements.

Figure 2.4: Images used by the Underworld project to convey the 3D complexity of buried pipes and cables. *Source: http://www.comp.leeds.ac.uk/mtu/farrimond.pdf*

- DNF will incorporate and adopt existing *de facto* and *de jure* standards, wherever they are proven and robust.

DNF, being a framework, does not include the data themselves. OS MasterMap is a data product which is based on DNF, its 'layers', all of which conform to DNF. Many of the data in these layers are owned by Ordnance Survey (such as the Topographic Layer and the Integrated Transport Network); others are owned and maintained by other companies using DNF (e.g. the UK Hydrology Office and the British Geological Survey). The value of DNF in this context it to allow other data providers to seamlessly integrate their data.

The concept of the TOID is part of DNF's specification, which can be applied to any of its layers. Blocks of TOIDs have been allocated to different organisations to ensure that duplicates are not allocated.

The Buried Services Working Group is looking at how DNF could be used for holding data on buried pipes (Cumberbatch, 2005; Cullen, 2005) and has brought together various utilities companies to discuss this. The Underworld Project[8] is a pilot scheme to map underground pipes and cables. Figure 2.4 shows the complexity of surveying and describing pipes, which potentially weave amongst each other and may contain sections of different ages.

The move to more structured feature-based digital mapping reflects the increasing importance of geographical information, rather than maps for simple display.

2.3.6 3D geometry and true feature-based databases

National mapping agencies do not usually provide national coverage of three-dimensional information except terrain surface models (derived from LIDAR, stereo aerial photography or similar). At first glance, this appears to be rooted in a very map-centric view of the world from which digital mapping has evolved. However, there are a number of obstacles to providing more comprehensive 3D data facing national mapping agencies in this regard:

- 3D data acquisition on national scales is a major undertaking, especially where it has to be intensively supported by ground-based surveying techniques.

[8] http://www.comp.leeds.ac.uk/mtu/

Figure 2.5: *a)* The outer surface of an urban area, represented as a point cloud using data collected by LIDAR) *Rendered in ArcScene using LIDAR data from Infoterra. b)* Viewshed analysis. The light-coloured areas form the visible field from the dot (which has been positioned 1.5 metres from the surface in a vertical direction). *Rendered in ArcScene using LIDAR data from Infoterra and 3D Analyst.*

- The term '3D data' can be applied to any data involving height which varies from 2D data annotated with feature height information to full 3D models where all three dimensions are described with the same priority (section 4.5.5).
- It is not clear how data would be distributed to customers and what proportion of these could deal with the 3D data as many customers use 2D software environments (this particularly applies to GIS software).
- True 3D data, over very large geographical areas, would result in vastly increased data volumes.
- In short, the business case has not yet been made.

Interest in 3D data (other than terrain models) is usually confined to man-made structures in an urban context, particularly buildings. However, the same concepts should be able to be applied across the real world.

Figure 2.5a shows an unstructured point cloud representing the upper surface envelope of an urban area. This simple 3D model can be easily acquired and can be used for a range of geometrical analyses (figure 2.5b shows the result of a viewshed analysis). Ratti (2002) demonstrated a series of analyses which can be performed on a raster version of such, including energy consumption in buildings, radiation exchange, air movement and space-syntax-based techniques.

For applications which require conceptualisations of real-world features, such surface-based models are inadequate, unless augmented by data on conceptualisations of real-world features. For example, OS MasterMap features could be spatially related to the surface (through a common georeferencing framework) in order to derive the 3D outer shell geometry of OS MasterMap features.

When only considering the outer external 3D geometrical shell of features (objects), features are still defined in 2D; e.g. a building defined as the structure above its footprint. This thesis argues that this does not add much extra information to the data resource. What would add give more potential for the use of the data is the

ability to define features in 3D. For example, instead of a feature being represented as the external 3D shell of the structure above a building footprint (corresponding to a 'building'), if 'room' and 'storey' features could also be represented. In this way, a 'building' feature can be decomposed into smaller features, horizontally, as well as vertically, leading to models of the *3D structure* of the built environment being produced. In this way, storeys, rooms and other functional spaces within the built environment in which human activity can occur can be captured and described. There are major obstacles with the practical implementation of this, almost entirely do to with data acquisition.

The eventual result of the new paradigm shift in cartography is expected to be a truly feature-based database. In such a truly feature-based database, the central concept is the *feature* (which in a 3D context is defined in 3D) with an identity (a persistent identifier such as OS MasterMap's 'TOID') and a set of attributes. The geometry of the feature is one of its many attributes. The feature-based database is underpinned by an explicit framework, in which multiple conceptualisations of real-world features can be defined and have their geometry described in three dimensions. The need for an explicit framework to support the automatic interpretation of the built environment is illustrated by the applications in section 2.9.2.

2.4 Boundaries

Tools for modelling geometry almost always assume that the data are known within a particular accuracy and precision. This is generally appropriate for the applications designed to be supported by computer aided design tools (CAD), because geometry is generally represented as designed. However, this may be less appropriate for applications designed to be supported by geographical information systems (GIS) because they are generally concerned with the *'as exists'* world in which data has to be collected, rather than the 'as designed' world where the data are already available. As well as the lower temporal certainty about the state of the 'as exists' world, boundaries may be ambiguous because they are dependent on feature conceptualisations. Also, some features do not have distinct boundaries Frank (1994b), particularly those which are natural features (Burrough, 1994). In the urban environment, the greater incidence of man-made structures leads to a greater number of features with distinct boundaries, although boundaries which do not correspond to physical features (e.g. postcode areas) are not necessarily distinct (Campari, 1994). Most tools which model geometry enforce precise geometrical boundary positioning, even if the boundary is indistinct or not known to the precision required. This includes GIS software, even though uncertainty is an acknowledged issue in GIScience. Frank (1994a) shows that in some cases, topological analysis is more important than the analysis and comparison of precise geometry. Section 2.4.2 also shows Ordnance Survey's practice of inferring an internal boundary of a building where it is known

that a portion fulfils a significant role; i.e. the fact that this portion exists is more important than its precise boundary delineation.

The surveying tradition works on the premise that everything can be surveyed to a given temporal and spatial accuracy and precision; clearly this has supported a long history of using plans for planning. Since there are no natural units of space, space has to be partitioned using observable boundaries. As mentioned earlier, many boundaries exist for different conceptualisations of features and subdivisions of these. The resulting features delineated by surveyed boundaries are unlikely to be universally applicable. This is immediately apparent when trying to match different sources of data; for example postal delivery points, land registration information, large-scale national mapping and local authority gazetteers. Different applications and agencies have different criteria for conceptualising features and defining their boundaries (section 2.9.2). It is unlikely that the data corresponding to the different feature conceptualisations can be easily related amongst features. A couple of examples will be presented in the following sections.

2.4.1 Land registries

The conceptualisation and extent of 'property' ownership units for land registries are based on legal definitions because they form parts of legal documents of ownership.

The *'fixed boundaries'* land registration tradition requires the absolute position of the boundaries of land or properties units to be fixed, such that they can be marked on a map. Registration is an expensive and time-consuming undertaking, because all owners of bounding land must settle any ownership disputes during the process of registration. Such fixed boundaries can easily be digitised to form the basis for a digital land registry database.

The *'general boundaries'* land registration tradition describes land and property boundaries using textural descriptions. These descriptions refer to visible features of the landscape (e.g. rivers, roads and hedges), which override map-based definitions; see figure 2.6 and the rather endearing comment in figure 2.6c. General boundary descriptions are prone to give rise to boundary disputes but may only need to be resolved for planning applications and ownership transfer. They are easier to map because they use visible features in the landscape[9]. Clearly, the concept of general boundaries is problematic from a geometrical modelling point of view and this makes the prospect of a full digital cadastre for England and Wales unlikely (Maynard, 2001). Legally, it is possible to fix an individual boundary numerically (the 'determining the boundary' process), using a qualified surveyor and the agreement of all owners affected (HMLR, 2004).

Most current practice in fixed-boundary definition is to define boundaries in two dimensions. In most countries, the right of ownership is implicitly assumed to extend

[9]In England and Wales, general boundaries were introduced (in 1875) because of the failure of the earlier 1862 Act requiring boundaries to be provided the nearest inch (Maynard, 1989).

a

Figure 2.6: Example of land parcel data from the land registry for England and Wales. *(a)* An example of a map of a land parcel from a 1968 conveyance; *(b)* Ordnance Survey Superplan Data (in grey) overlaid with the boundary shown on the HMLR filed plan (red) and the author's survey (blue); *(c)* Warning note appearing at the foot of the conveyance plan used in 1978 for the registration of the property shown in (b). *Source: Maynard (2001)*

from the centre of the Earth to the sky; for example the Netherlands, Israel and Sweden (Stoter, 2002; Shoshani *et al.*, 2005). In Sweden, permission to construct a tunnel below a property needs the property owner's permission (Julstad and Ericsson, 2001). In Norway, ownership is considered to extend down as far as it can be 'utilised', thus outside the central business district, tunnels more than a few metres below property are not considered to impinge on ownership (Valstad, 2001).

However, particularly in multiple-occupancy buildings in urban areas, one two-dimensional land parcel may have multiple uses (section 2.3.6. van der Molen (2001) lists examples:

- Multi use on ground level – at different times of day, land may have different uses.
- Multi use above surface – flats on multiple storeys
- Multi use sub-surface – underground shopping malls, roads, tunnels and pipelines
- Multi use time-dependent – an EU time directive allows time-shared ownership.

The two middle uses (above) are common practice and the legal systems of the majority of countries allows for the concept of the ownership of flats within a block of flats on different storeys. In countries whose digital cadastre is populated with 'fixed boundaries', there is considerable interest in how to represent the 3D legal situation (this is the subject of a PhD thesis by Stoter, 2004). Within current practice, digital cadastres have had to find ways of registering multiple occupancy within a land parcel including multiple storeys within a 2D legal and technical framework. In the Netherlands where fixed boundaries are used, 3D situations are

Figure 2.7: Example of an apartment complex (left) and an overview of the whole street (right). *Source: (Stoter and Ploeger, 2003, figure 3, p558)*

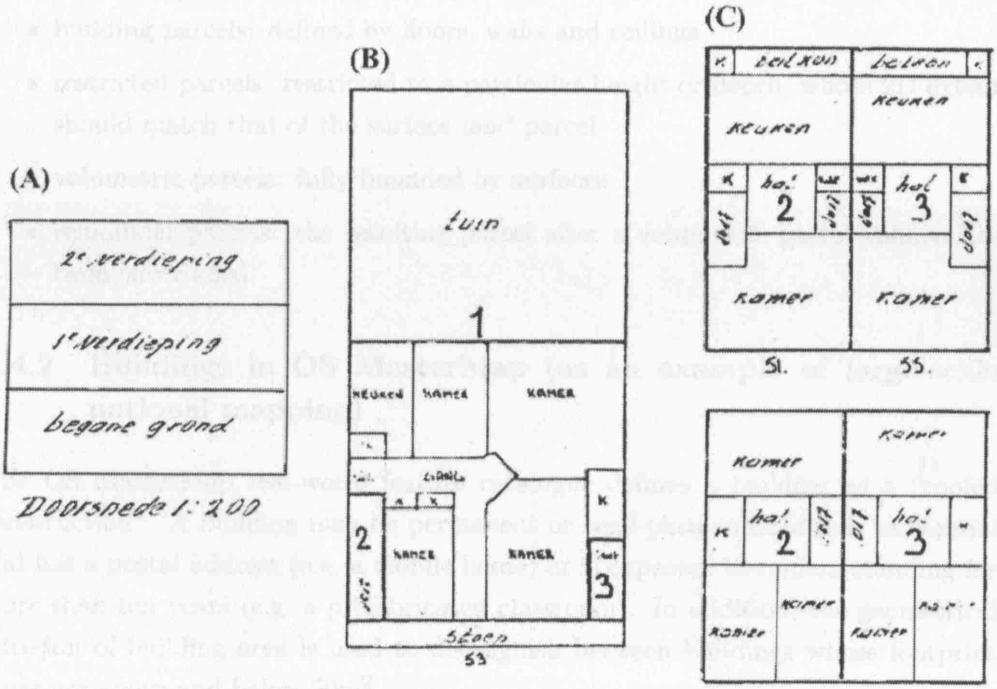


Figure 2.8: Drawing belonging to the deed of division of the apartment complex in figure 2.7. (A) cross-section; (B) first floor; (C upper) second floor; (C lower) third floor. The individual apartments are indicated by 1, 2 and 3. *Source: (Stoter and Ploeger, 2003, figure 4, p 559)*

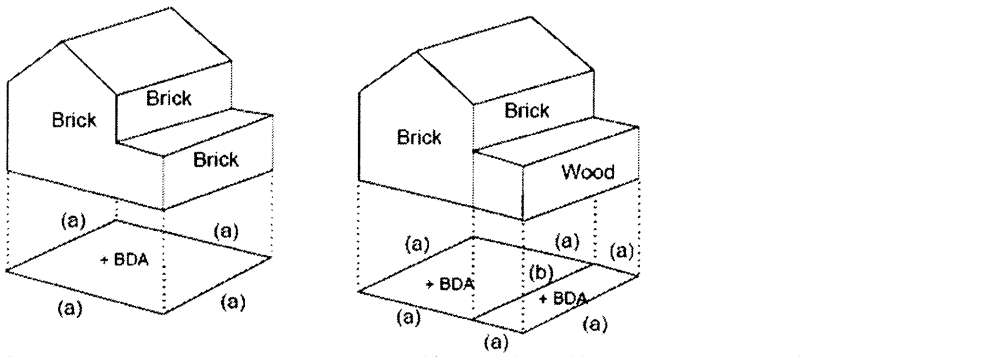


Figure 2.9: A building is defined as a "roofed construction" by the OS MasterMap real-world feature catalogue. This figure shows how a structure may be subdivided using physical characteristics observable from outside, including the construction material. *Source: Ordnance Survey (2001, p75)*

attached as supporting documentation (figure 2.8); thus are held separately from the main spatial data. In Norway a similar approach is used (Valstad, 2001).

In Queensland (Australia), property registration in three dimensions is already commonplace (van Oosterom *et al.*, 2006). This was introduced to account for multilevel ownership. Four types of parcel with a height component are distinguished (Stoter and van Oosterom, 2006, p54-55):

- building parcels: defined by floors, walls and ceilings
- restricted parcels: restricted to a particular height or depth, whose 2D extent should match that of the surface land parcel
- volumetric parcels: fully bounded by surfaces
- remainder parcels: the resulting parcel after a volumetric parcel volume has been subtracted.

2.4.2 Buildings in OS MasterMap (as an example of large-scale national mapping)

The OS MasterMap real-world feature catalogue defines a building as a "roofed construction". A building may be permanent or semi-permanent if it is residential and has a postal address (e.g. a mobile home) or is expected to remain standing for more than ten years (e.g. a prefabricated classroom). In addition, the geometrical criterion of building area is used to distinguish between buildings whose footprint areas are above and below $50m^2$.

Adjacent building parts are captured as separate buildings if they are made from a different building material, as shown in figure 2.9. In addition, if a roofed structure has an 'important' office within, this is captured – as illustrated in figure 2.10 – even if the boundary is not observable from the outside.

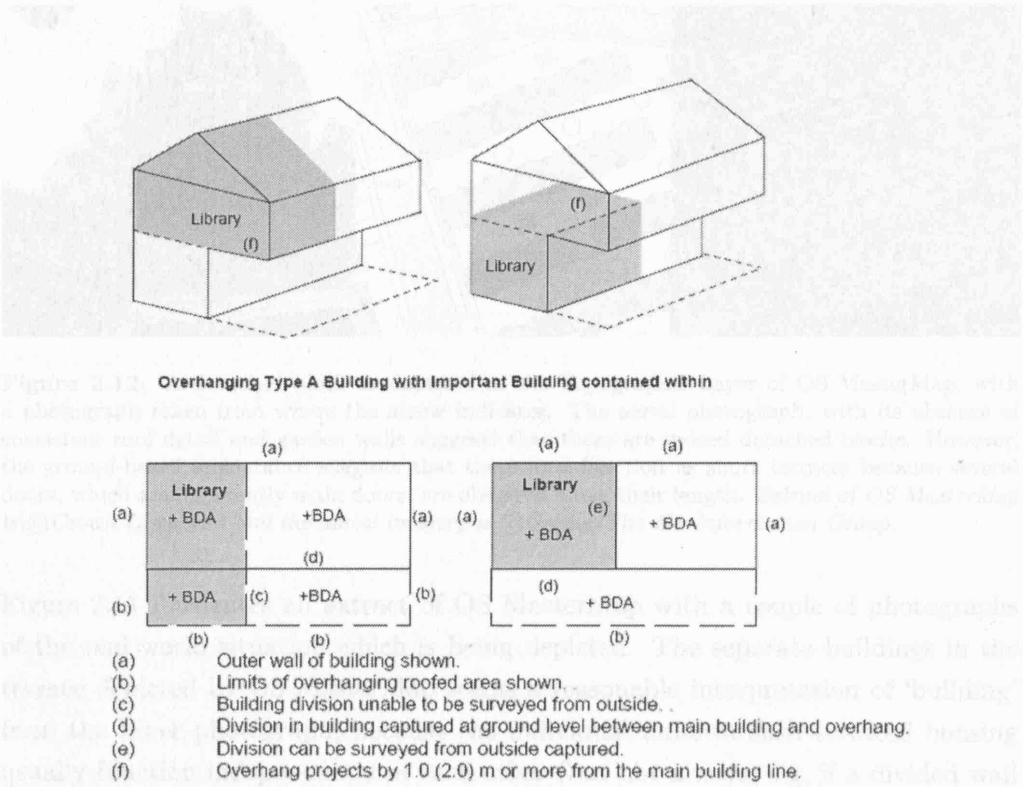


Figure 2.10: An 'important' office within a roofed construction is captured for OS MasterMap, whether or not the internal division is directly observable from the outside. *Source: Ordnance Survey (2001, p79)*

Figure 2.11: A terrace of housing as depicted in the Topographic Layer OS MasterMap, accompanied by photographs and aerial photographs. The photograph appears to be consistent with the divisions in the terrace as shown in the map. The internal division, although not directly observed can be inferred from roof detail and garden walls. The upper photograph shows an inconsistency with the maps with two 'buildings' and garages depicted as one. *Extract of OS MasterMap is ©Crown Copyright and the aerial imagery is ©Google/The GeoInformation Group.*

Figure 2.12: A detached block as depicted in the Topographic Layer of OS MasterMap, with a photograph taken from where the arrow indicates. The aerial photograph, with its absence of consistent roof detail and garden walls suggests that these are indeed detached blocks. However, the ground-based appearance suggests that these may function as short terraces because several doors, which are apparently main doors, are observed along their length. *Extract of OS MasterMap is ©Crown Copyright and the aerial imagery is ©Google/The GeoInformation Group.*

Figure 2.11 illustrates an extract of OS MasterMap with a couple of photographs of the real world situation which is being depicted. The separate buildings in the terrace depicted by OS MasterMap seems a reasonable interpretation of 'building' from the lower photograph, because the individual units in such terraced housing usually function independently of each other (but not always; e.g. if a divided wall is removed). Aerial photograph interpretation is likely to have been used and the aerial photograph in the figure shows that roof detail and the presence of garden walls can be successfully used to infer the likely subdivisions of the terrace. The upper photograph suggests that the corresponding building as depicted by MasterMap may in fact be two buildings (one of which has a set of garages below it). The situation from the air is obscured by a tree which may have contributed to the problem.

Figure 2.12 shows a set of buildings which have been depicted as a series of detached blocks. These have a number of main access doors with buzzers to different flats, thus it is possible that these function as short terraces, but there is little or no externally-visible detail to suggest this.

These two examples illustrate the problems with the real world definition of buildings. From the outer appearance, interpretations can be made, but behind the building façade, the situation may be very different. For example, rather than demolishing structurally unsound listed buildings, the structure and interior may be replaced whilst the façade is retained.

So in OS MasterMap, buildings are defined by geometrical means, but also by functional means using inferred internal boundaries if necessary (figures 2.11 and 2.12).

The non-domestic building stock project (section 2.9.2.3) defines building units in a similar fashion, ensuring that each building unit has a common value for each of the characteristics surveyed from the street. Section 4.6.1 further discusses the ways in which buildings can be defined.

The National Land and Property Gazetteer (NLPG; section 2.9.2.1) allows different types of units to be defined are shown in table 4.1, as defined by British Standard BS7666 (section 4.6.5).

2.4.3 Time and boundaries

The delineation of boundaries represents a snapshot in time and the units which the boundaries delineate are liable to change. The time, nature and frequency of the change are dependent on the stability of the units. Many boundaries are not usually temporally stable. Inherently fuzzy natural boundaries (e.g. rivers, weather systems) may divide and disperse over time and then re-form later (Hornsby and Egenhofer, 1997). The visible boundaries of man-made structures are usually temporally stable; they are not subject to significant erosion and are usually designed to persist for decades. Changes such as the construction of a new extension onto a building are usually sharply defined. However, the units of ownership, occupancy and activity within land parcels, can change independently of the structure's geometrical fabric. As with natural phenomena such as weather fronts, units may join (e.g. destruction of an internal boundary in order for two units to function as one) and then separate again (e.g. reinstating the internal boundary).

The history of transactions which affect land ownerships is recorded by land registries and this provides the opportunity for spatio-temporal querying and reasoning. This fact was recognised by Al-Taha (2001) who reviews and explores temporal reasoning for land registries. This includes support for distinguishing between different time perspectives, particularly the creation time (when the action took place) and the recording time (when the action was recorded in the system). This issue was also addressed in chapter 11 of Stoter (2004). OS MasterMap also records changes which are used for supplying change-only updates to customers. However, such information is just a by-product of the update process rather than an attempt to support spatio-temporal queries.

2.5 Georeferencing

A framework is essential in order to georeference and to define a position or feature in space. Sections 2.3.1, 2.3.5.1 and 2.3.5.2 all refer to the georeferencing systems used by national mapping organisations in general and Ordnance Survey in particular. The National Grid used by Ordnance Survey is a Cartesian coordinate system. OS MasterMap additionally introduced discrete georeferencing, where existing features can be identified by a unique identifier (the 'TOID'). Another popular approach which is used for the US Census is a *network segmentation* approach which is based on a geometrical network and assumes that the feature to be addresses lies along a part of this network. The feature's position is indicated by the network link on which it lays and the distance along the link at which it is positioned. Lee (2004b) adapts

this similar scheme for georeferencing within buildings by building a 3D geometrical network of corridor centrelines and identifying rooms as distance ranges along the corridor network segments.

Postal addresses are also important for the discrete georeferencing of data. The human-readable textual nature of postal addresses gives rise to range of address matching errors, which are due a lack of consistency in human-readable textual addresses (Pun-Cheng and Lee, 2004). The National Land and Property Gazetteer (NLPG, which is described in section 2.9.2.1) has an address module which allows alias and multiple addresses to be defined for properties. This is illustrated in figure 2.18. Since postcodes are more formalised, there are fewer problems with postcode matching, but problems still remain. Postcodes are generally designed for the sorting and delivery of post rather than for georeferencing for GIS-type purposes and there are postcode changes every year. Nevertheless, their widespread use, including for many spatial statistics, maintains their importance for geospatial information.

2.6 Tools for handling geometrical and geographical data

2.6.1 Computer aided design (CAD)

Computer aided design (CAD) packages are general-purpose design tools for engineers. The first recognisable CAD prototype was SKETCHPAD in 1963. In the 1970s CADs were simply 2D geometrical drawing packages which could be used to edit, store, update, print and distribute 'digital drawings'. Development continued throughout 1970s when the benefits of computers were realised for dealing with the geometrical aspects of engineering projects. Research focused on how computers could be used to create accurate (2D) digital drawings, continuing to use drawing conventions such as cross-sections and exploded drawings.

Version 1 of AutoDesk's AutoCAD was released in 1982. With it came the DXF file format which was intended to allow data interoperability between AutoCAD and other software packages, being an exact representation of the data in its closed native file format, DWG[10]. This was widely supported by all major CAD packages eventually becoming a *de facto* standard. The file format of early versions of DXF (point and line-based geometries with group codes, arranged in 'layers') provides a good insight into early CAD packages. The 'layers' were used to add structure to the data, organising groups of geometries into themes. Conventions for naming layers are defined by ISO 13567. Figure 2.13 shows a DXF floorplan. The geometrical elements shown have no 'intelligence' about what they represent. Lewis (1996) and Lewis and Séquin (1998) describe a system for interpreting such floorplans, but such systems are prone to error.

[10]http://en.wikipedia.org/wiki/DXF

Figure 2.13: Floorplan for part of one of UCL's buildings, including part of a car park and a ramp which goes underground. *Source: Estates and Management, University College London.*

CAD also developed more advanced geometrical modelling capabilities. The use of 3D coordinate systems allowed *wireframe* models to be represented. Later, surfaces were added: *parametric surfaces* originally designed for modelling the curved panels used for the construction of aircraft and cars and *boundary-representation* models which represent solids as closed sets of surfaces. These approaches implicitly model solids using their boundaries; however, the 1980s saw the development of solid models (Mäntlyä, 1988) which were able to represent the solids themselves by combining geometrical solid primitives with the 'union ($\cup$)', 'intersection ($\cap$)' and 'difference ($\setminus$)' operators (see section 4.5.3.2). The development of full 3D models made it possible to produce any number and manner of consistent cross-sections, projections and perspective views from any angle.

In architecture, geometry-based design has been summed up in the sentence "traditional building design and documentation methods using conventional CAD emulate a drafting pencil electronically [resulting in] digital drawings that are just a collection of lines and arcs that do not represent the real world equipment and materials that make up a [building]" (AutoDesk, 2003, p2). 'CAD' as used in this thesis will refer to the geometry-based design approach described above. Section 2.6.4 describes application-specific parametric modellers which assist in the design process by allowing the specification of design constraints. As will be seen, such application-specific

approaches have had a long history in research, but have now entered the mainstream software market; AutoCAD recently bought parametric building modeller Revit® which it is marketing as a mainstream piece of building design software.

2.6.2 Geographical information systems (GIS)

GIS have some of their roots in CAD because at the time of their early stages of development, CAD software was the only family of software tools capable of handing vector geometry. However, their paths of evolution diverged, producing a new family of software in which analysis of the 'as exists' world could be performed by combining geographical data from different sources, analysing the topology of the imported data and using feature-based representations corresponding to real-world features. While CAD developed into increasingly sophisticated geometrical modellers, GIS concentrated on spatial analysis technology for 'real-world features'. GIS provides tools to describe real-world features using an abstracted geometrical representation and a set of non-spatial attributes, and topology models for building and for describing spatial relationships between features.

2.6.3 Comparison of CAD and GIS

The differences between the software systems of GIS and CAD have been widely observed e.g. Cowen (1990); Newell and Sancha (1990); ESRI (2002). Although comparing and integrating CAD and GIS models is not the subject of this thesis, some of the issues associated with this are relevant.

In terms of the *application domain*, CAD packages are aimed at precisely representing the geometry of 'as designed' objects, whereas GIS are aimed at abstracting the real world into features upon which spatial and non-spatial analysis can be performed. There are also differences in scale – scales of geometrical representation and of feature conceptualisation. Despite these differences, both families of software are often used to deal with buildings, but these are dealt with at different abstractions and scales.

In terms of *technology*, the gap between CAD and GIS is closing because advances in interoperability and computer hardware capabilities are producing technical solutions which combine aspects of both systems. Some of the literature tends to overstate the differences (in some cases, arguably, to justify a research agenda). However, there is no doubt that the design and capabilities of GIS and CAD software system reflect the different emphases of the application domains they are aimed at supporting.

In terms of *data volume*, GIS datasets tend to be very large because of the vast geographical extent. In CAD-type applications, even though the scale and geometrical detail of the phenomenon being modelled is much greater than for GIS-type applications, the geographical extent is much smaller. This is one of the reasons for

not modelling high geometrical detail, especially since such geometrical detail is not required for GIS-type applications. (Another reason is the difficulty in obtaining such information over such large geographical extents).

Theoretically, however, there are differences which are yet to be resolved. A good discussion of the issues can be found in Zlatanova and Prosperi (2006). The issues are mainly to do with the lack of ontological interoperability due to the differences in uses (application domains), differences in scale leading to different scales of 'features' being of interest and the different natures of the data being modelled. The former two points can be solved by the design of ontological models to match concepts across different scales and applications.

2.6.4 Domain-specific parametric modellers

Parametric modellers are feature-based modellers which use parameters to specify geometrical and other design constraints which are dependent on the *function* of the various features. The domain reviewed in this section is building construction, and it shows the flexibility offered by feature-based approaches and the increased potential for automated computer interpretation offered by the underpinning framework.

The building construction industry has a long history of using computer modelling. Since large building projects are usually one-offs, there has been a tradition of using rather *ad hoc* techniques for building digital representations of planned constructions for different uses. As an architect from Benoy, a London-based architectural firm, commented, architecture is often "still a cottage industry"[11].

2.6.4.1 Design assistance

There were early attempts to specify constraints on spatial configuration to assist in the design of mass construction projects in the mid-1970s (Eastman, 1999). In Britain, OXSYS CAD was developed for the National Health Service to assist in the design of new hospitals using predefined building components. CEDAR and HARNESS were used for hospital design. The Scottish Special Housing Authority developed a system for assisting in the design of council houses and the layout of housing estates. In the United States, Techcrete, ARCH-MODEL, GLIDE and GLIDE-II are examples of similar projects. These attempts allowed problems of packing and spatial configuration to be described. These were used to constrain algorithms for generating building plans.

These projects were limited by the computer hardware capabilities, they were very specific for particular building types and so could not be easily applied to other similar projects (Eastman, 1999).

[11] James Utting, personal communication (December 2003).

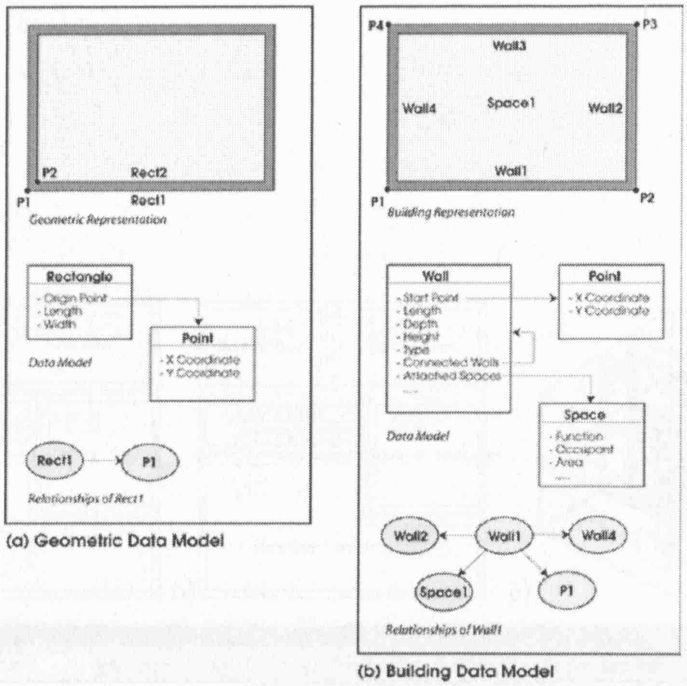


Figure 2.14: Simplified illustration of the difference between a geometrical model (pure geometry) and a building model (feature-based with conceptualisations of those features relevant to buildings). *Source: Khemlani (2004).*

2.6.4.2 Unified building models

As discussed in section 2.6.1, general-purpose CAD systems are geometry-based modellers. Support has been added for application-specific libraries of 'features' composed of collections of geometries; e.g. AutoCAD's domain-specific extension Architectural Desktop (Khemlani, 2002). However specialist building modellers are more effective at conceptualising tangible parts of buildings, their characteristics and the constraints they impose on other building parts. Figure 2.14 shows, in simple terms, the difference between geometrical modellers and building modellers (or more widely, feature-based modellers).

There have also been various academic projects which have aimed to produce flexible and multi-purpose building modellers. The Building Design Advisor (BDA) Project (Papamichael *et al.*, 1997) was initially set up (in the 1980s) to do research into the involvement of computers in the design of more energy-efficient buildings. By 1994, the project goals became focused on the design of a unified data management system for buildings, capable of supporting a wide range of external simulation tools (Papamichael *et al.*, 1997). At the time, CAD systems were widely used for building modelling, but poorly linked to analysis models (such as pedestrian modelling, smoke spread and heat loss). It was difficult to export data to such external tools and as a result, they were underused. Figure 2.15 presents screenshots of the graphical user interface of the prototype, which illustrates the hierarchical, feature-based view of the building. The data model aims to support a wide range of applications, rather

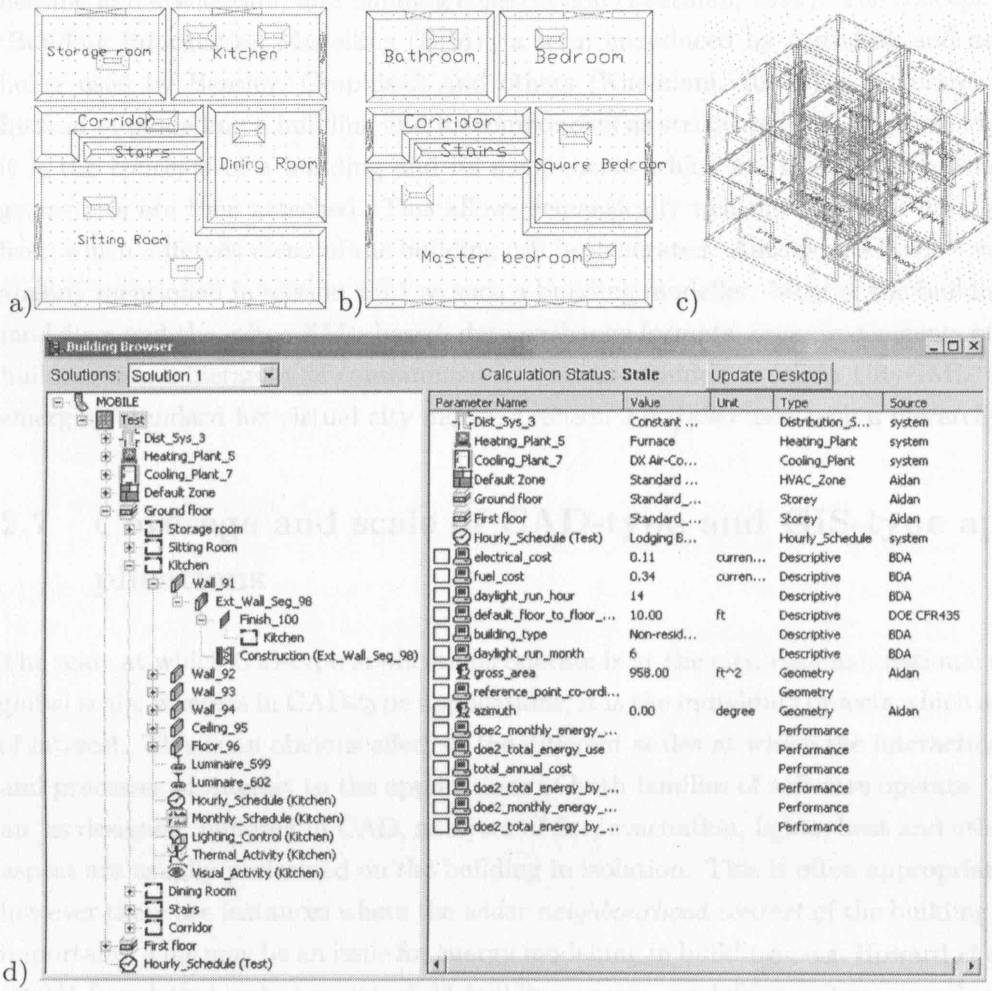


Figure 2.15: *(a)* Floorplan of the ground floor, complete with windows (on the rooms' edges) and lights (within the rooms). *(b)* Floorplan of the first floor. *(c)* Perspective view of the building. *(d)* The BDA Building Browser. In the left plane the components of the building are arranged in a hierarchy, each component being an object. The building is called "test" here (highlighted), which contains two floors. The ground floor contains the various rooms, the rooms contain walls and the walls contain wall segments, one for each side. In the right-hand pane, the attributes of the highlighted object are listed.

than being optimised for geometrical representation. Each of the components is an object, which has parameters, relations to other objects and methods which describe its behaviour.

More recently, there have been initiatives in the construction industry to develop the concept of a unified model to support all stages of project management including the initial specifications, geometrical models, analytical models (e.g. loading, lighting, heating and evacuation) and building construction (Eastman, 1999). The concept of 'Building Information Modelling (BIM)', a term introduced by Autodesk and now being used by Bentley, Graphisoft and others (Khemlani, 2003b) is mainstream. Instead of modelling a building and its components as structured sets of geometries, it is the *concepts* of a building and its components which are modelled, to which geometries are then attached. This allows semantically rich models to be created, from which different *views* of the building can be generated. AutoDesk's Revit® was already mentioned in section 2.6.1 as such a building modeller. Most of the building modellers and the, often XML-based, data exchange formats, organise elements of a building into a hierarchy of containment in a similar fashion to BDA. CityGML, an emerging standard for virtual city models (section 2.9.1) also uses such a hierarchy.

2.7 Coverage and scale of CAD-type and GIS-type applications

The scale at which GIS-type applications operate is at the city, regional, national or global scale; whereas in CAD-type applications, it is the individual objects which are of interest. This is an obvious effect of the different scales at which the interactions and processes of interest to the applications of both families of software operate. In an 'as designed' building in CAD, analyses of fire, evacuation, lights, heat and other aspect are usually performed on the building in isolation. This is often appropriate, however there are instances where the *wider neighbourhood context* of the building is important. This may be an issue for energy modelling in buildings; e.g. Howard *et al.* (1994) found that only ten out of 33 building energy modelling software products took into account overshadowing effects of other buildings. In terms of accessibility metrics and evacuation, the *connection of the building's access routes with routes outside the building* is also important (section 2.9.4).

A solution is to embed from CAD detailed individual building with GIS data. Stoter and Ploeger (2003) found that there are practical difficulties, due to the following reasons:

- Contractors and builders still widely use 2D.
- Such plans have much more geometrical detail than needed for the cadastre so need to be generalised and would then have to have the boundaries reassessed for legal registration.

- Coordinates need to be transformed from a local system to the national geo-referencing system.
- CAD files quickly become large and unwieldy; they are flat files optimised for rapid on-screen display.
- It is difficult to produce the spatial primitives required automatically (CAD files tend to be relatively unstructured in the same way as Land-Line (section 2.3.5.1).

2.8 Classifications and feature catalogues

Classification schemes are used for grouping similar features together on maps (as described by map legends) and in feature-based models. In feature-based representations, classification schemes, feature catalogues and dictionaries for relating semantic terms are necessary in order to be able to identify comparable feature concepts, even across large areas. For example, OS MasterMap has a real-world feature catalogue which describes its classification (Ordnance Survey, 2001) used for features over the entire area of Great Britain.

CAD-type applications are usually only concerned with comparing concepts within an individual building, rather than between buildings. However, the increasing importance of dedicated parametric modellers for buildings to support various stages of the construction process, necessitates a standard for defining and identifying features so that they can be compared with other building models and with other analysis tools. Standards and frameworks for the building construction industry are reviewed by Eastman (1999).

The ISO-STEP standard (ISO 10303, Industrial Automation Systems - Product Data Representation and Exchange) as defined by the International Standards Organisation[12], defined standards used by the building construction industry. Part of this standard is a lexical language called 'EXPRESS'(ISO, 1994) which provides a formal means for defining data, data relationships and the constraints necessary to describe a product (Wilson, 1998).

The Alliance for Interoperability (IAI) is a consortium of industrial partners including CAD vendors. In 1997, they released the Industry Foundation Classes (IFC), defined using ISO-STEP (and EXPRESS). IFC is a data exchange format for describing buildings and the elements of which they are composed. The description includes the 3D geometry, the semantics of the elements, the relationships between elements and attributes for these elements. The semantic richness is intended to support a role of a Building Information Model, in which it is possible to automatically transform the data to different abstractions and according to the specifications and needs of a range of modelling and analysis software tools.

[12]http://www.iso.org

Some objects of interest are common to both GIS-type applications and CAD-type applications, the most obvious example being the 'building'. Section 4.6.1 illustrates the different ways in which buildings can be conceptualised and subdivided. The mismatch between building units from different data sources for different application domains also illustrates that although similar features are of interest, the precise mapping of these concepts is difficult. Such a mismatch might be resolved by using semantic dictionaries and ontological frameworks. However, since there are no equivalents of most of the IFC concepts in MasterMap (this is obviously the case, because IFC deals with buildings and their interiors whereas national mapping deals with exterior space), the transformation of IFC data into MasterMap would lose almost all the detail of IFC's building model. This thesis argues that not only do spaces within buildings in 3D need to be represented, but that there is (potentially) an infinite number of feature conceptualisations which must be definable. Thus, the framework design proposed in this thesis for a national mapping aims to support both these; i.e. to be able to represent the spaces within a building and to be able to flexibly define different features of various spatial extents.

2.9 Potential applications of large-scale digital mapping in 3D space

This thesis proposes that a 3D national mapping framework is needed which is capable of providing flexible 3D feature-based data for a variety of applications. This section will review examples of applications which would benefit from such a framework. The characteristics of these applications are that they are concerned with specific conceptualisations of features which describe the 3D structure of the environment. This includes decomposing buildings into units within buildings, on different storeys.

2.9.1 Virtual cities

The term 'virtual city' is a rather vague term, but in this context, it is used to refer to a digital representation of a real city for use in a wide variety of contexts (Batty *et al.*, 2001; Hudson-Smith and Evans, 2003; Kolbe *et al.*, 2005). Implementations vary widely, but they are usually based around a 3D geometrical representation which is used applications of urban planning and social issues. Using disaster management as an example, a virtual city could be used to keep a record of the state of a city before a disaster for the purpose of reconstruction, provide 3D visualisations, provide the input for navigation for augmented reality systems, identify escape routes and develop flooding scenarios (Goodchild, 2006).

Figure 2.16: Examples of 'virtual cities'. *(a)* Aerial photo draped on a digital surface model of part of London, with a 3D model of St Paul's Cathedral embedded. *Source: Skyline's TerraExplorer.* *(b)* Buildings modelled as extruded polygons with a detailed 3D building model embedded. *Source: Clover (2000) c)* Screenshot of the 'The Getaway' game for the Sony Playstation.

Hudson-Smith and Evans (2003) provide a worldwide review of virtual cities. These range from the wholly geometrically simple to those which incorporate geometrically detailed elements.

Some virtual cities have an emphasis on graphical display. Those which consist of maps combined with some height data and perhaps with aerial photography (e.g. the data model used by TerraExplorer[13], output of which is shown in figure 2.16a) allow little more than fly-through visualisations. Figure 2.16c is more visually realistic – it shows part of photo-realistic model of London used in the computer game, 'The Getaway'. This model supports the rapid generation of 3D visualisations from a restricted set of viewpoints and includes the attributes and topology required for the gameplay. The buildings appear to be well-modelled, but are based on relatively simple geometrical forms with high quality texture mapping to give a high apparent level of geometrical detail. Since the model is only designed to be viewed from the road level, the underlying geometrical model need not be as complete and complicated is if it had to support being viewed from all angles.

The US Government Institute for Defense Analysis (IDA) is running a project which aims to produce a digital city for interactive training in dealing with major terrorist threats (Clover, 2000). The model is a simple extruded polygon GIS for the major-

[13]Produced by Skyline Software Systems, TerraExplorer is available at http://www.skylinesoft.com/corporate/technology/technology_TerraExplorer.asp

Table 2.1: Levels of Detail for the Miller-Hare Model of London. *Source: Batty et al. (2001)*

Object Type	Definition
A	Detailed architectural model
B	Detail equivalent to 1:100 measured building survey
C	Detailed elevation
D	Major details of building elevation
E	Accurate building volume
F	Roofscape
G	Prismatic block models – coarse massing

ity of buildings. Amongst these, two types of detailed individual building models exist: building 'shells' which only represent the exteriors and are obtained from a third-party, and detailed buildings which have the interior modelled as well as the exterior for significant or public-use buildings (see figure 2.16b).

The London Travel Demonstrator Model is based on a 3D model of London which was originally developed for the COVEN project[14] and is concerned with immersive, multi-participant environments. It has been developed for applicability to tourism, allowing people to experience and rehearse travel though London before they arrive and access tourism-related data. It is a proof-of-concept model for part of London, rather than a comprehensive implementation (Batty *et al.*, 2001). The digital model of the city is a 2.5D model, using Ordnance Survey 2D footprints and height data from the Cities Revealed[15] dataset. Buildings are split horizontally into two or three divisions with fixed heights (entrance, middle and optional upper level) and each has a roof (Steed *et al.*, 1999). A few buildings have been modelled to a high degree of detail. For example, the Pearson building at UCL has its interior modelled to a high level of detail, with 100 rooms, some of which contain furniture.

Miller-Hare[16] is a small London-based consulting company who produce city visualisations to support the planning and marketing of new buildings. They have built a model of London using data collected over the past fifteen years. The advances in technology over this time have led to a rather *ad hoc* development path for the model with seven level of details, as seen in table 2.1 (Batty *et al.*, 2001). As in the IDA model, most of the buildings are 2.5D prismatic blocks. As various projects undertaken by Miller-Hare have been completed, so the model has been steadily upgraded.

The Virtual London project[17] is based at the Centre for Advanced Spatial Analysis at University College London. Its requirements are that it should not rely on detailed architectural drawings and that it should be able to reach as wide an audience as possible, ideally through the Internet (although this is not currently possible due

[14] http://coven.lancs.ac.uk/
[15] http://www.citiesrevealed.com/
[16] http://www.millerhare.com/.
[17] See article from GIS Professional, issue 1, November-December, 2004, pages 22-24

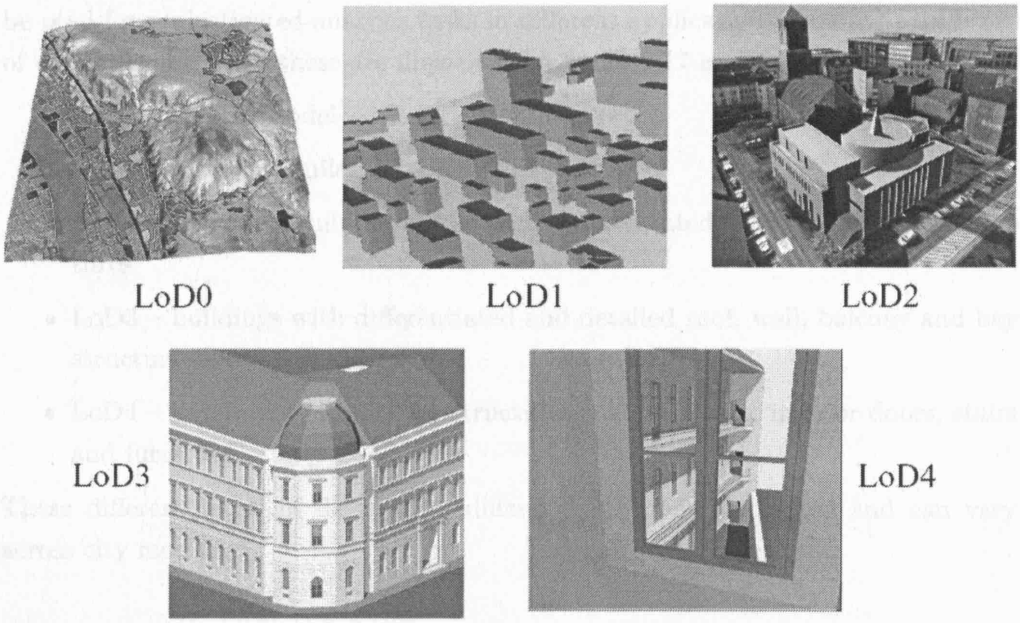


Figure 2.17: The five levels of detail defined in CityGML. Source: Kolbe *et al.* (2005).

to licensing restrictions). It covers the area within the M25 motorway (an increase in area since the cited article was published). The majority of buildings in the model are based on OS MasterMap's building features, with their heights taken from LIDAR-derived data, but a proportion of the most well-known buildings have been photogrammetrically rendered and a small fraction have photo-realistic rendering. This balance of detailed and simple 3D geometry has been chosen to help reduce the bandwidth required for planned distribution over the Internet. 3D panoramic views have also been embedded, allowing a user to see 360° views at particular locations. The choice of buildings to render in detail and the presence of 3D panoramic views is clearly aimed at visualisations, but the model has also been used to show the effect of sea-level rise, levels of pollution and socio-economic surfaces and how these correlate to the physical fabric of the built environment.

Implementations of 'virtual cities' vary widely. Some require the simplified 3D geometry of buildings (when compared with individual CAD models) across large areas and require these buildings to be distinct features which are classified in some way. Some virtual cities may require that different storeys to be treated as separate features.

Although 'virtual cities' vary in character, a standard for their description and exchange is emerging. CityGML[18] in an open standard for the representation and exchange of data on 3D urban objects, based on XML (Kolbe *et al.*, 2005). It is not just the geometry of the urban objects which are dealt with, but also topological information, semantic information, information on appearance and generalization hierarchies. The aim is to provide a framework which enables virtual city models to

[18]http://www.citygml.org/

be used for sophisticated analysis tasks in different application domains. Five levels of detail are included; these are illustrated in figure 2.17 and listed below.

- LoD0 – terrain model
- LoD1 – extruded building blocks
- LoD2 – extruded building blocks with differentiated roof structures and textures
- LoD3 – buildings with differentiated and detailed roof, wall, balcony and bay structures.
- LoD4 – buildings with interior structures such as rooms, interior doors, stairs and furniture.

These different levels of detail have different precisions associated and can vary across city models.

2.9.2 Inventories of building and property units

There is a wide range of applications for which data are attached to particular conceptualisations of units of buildings. These are the bases of spatial databases. The precise geometry of these units is not always essential for the applications, but usually the location is. Applications of inventories of building units range from simple data storage and retrieval of data (e.g. NLPG in section 2.9.2.1) in order to analyse building characteristics (e.g. patterns of commerce, activity or energy use) over large areas.

These applications demonstrate the need for large-scale national mapping which allows descriptive data on a range of building units which exist in 3D space to be stored and retrieved. This requires information about the units of buildings, but not at the level of detail of architectural CAD models. Crucially, they must be able to be retrieved in a way which allows them to be directly compared over large coverages.

2.9.2.1 National Land and Property Gazetteer (NLPG)

The National Land and Property Gazetteer (NLPG) is an implementation of a gazetteer based on BS7666 (section 4.6.5). It allocates a unique identifier to land parcels to which attributes can be added. This resource is used by councils and governments for applications such as land registration, the analysis of land use and property management (British Standards Institute, 2000b, Annex B). NLPG forms the core of the National Land Information Service (NLIS and ScotLis) initiative to "promote electronic delivery of land and property related information to a wide audience"[19]. The gazetteer also includes addresses. A standard address gazetteer is important for organising and coordinating nationwide surveys such as the collection

[19]http://www.nlpg.org.uk/_public/sheet2.htm

of data for the UK Census and maintenance of the Electoral Roll. It is widely accepted that the 2001 UK Census missed sections of the population due to a lack of a definitive database of property[20].

It is local authorities which are responsible for creating a section of gazetteer for their area, known as a Local Land and Property Gazetteer (LLPG). LLPGs are used by local authorities to support the administration and maintenance of their local areas. Many local authorities have websites which allow residents to view maps of their council area, showing the status of planning applications and the locations of council services; e.g. the Royal Borough of Kingston-Upon-Thames[21]. BS7666 defines the 'basic land and property unit (BLPU)', the fundamental units of NLPG, each of which has a 'unique property reference number (UPRN)' which is similar in concept to the 'TOID' in OS MasterMap. Primary BLPUs (e.g. a block of flats) can contain secondary BLPUs (e.g. flats). Figure 2.18 shows that each BLPU has an address (a main address and alias addresses). Kingston-upon-Thames' LLPG describes the geometry of each primary BLPU with a polygon corresponding to its footprint, which can be seen in figure 2.18.

The NLPG is an aggregation of all LLPGs. Local authorities submit the data to an organisation called 'Intelligent Addressing' who validate and clean the data, ensure all are comparable in quality, ensure that it can be matched to other national datasets (the Electoral Registers, Address-Point, Council Tax registers and Non Domestic Rating Lists[22]). It is then incorporated into NLPG. In the current implementation, geometry is stripped out; thus NLPG does not have any extents recorded[23]. Though the implementation of NLPG does not record any geometry, geometry is clearly of interest and is usually an important component of the local authorities' LLPGs. BS7666 allows the optional use of 2D polygons to describe extent, but no 3D information.

OS MasterMap features and BLPUs do not necessarily correspond with each other, because their respective features of interest are different. OS MasterMap features (in the Topographic Layer) define topographic detail, whereas NPLG is interested in "area[s] of land, property or structure of fixed location having uniform occupation, ownership or function" (British Standards Institute, 2000b, p1).

2.9.2.2 Land and property registration

There are many variants of land and property registration systems throughout the world (Bogaerts and Zevenbergen, 2002). The bounded units of interest are legally defined and form parts of legal documents.

[20] e.g. http://news.bbc.co.uk/2/hi/uk_news/2277835.stm
[21] http://maps.kingston.gov.uk/.
[22] http://www.nlpg.org.uk/_public/sheet4.htm.
[23] http://www.nlpg.org.uk/_public/sheet6.htm.

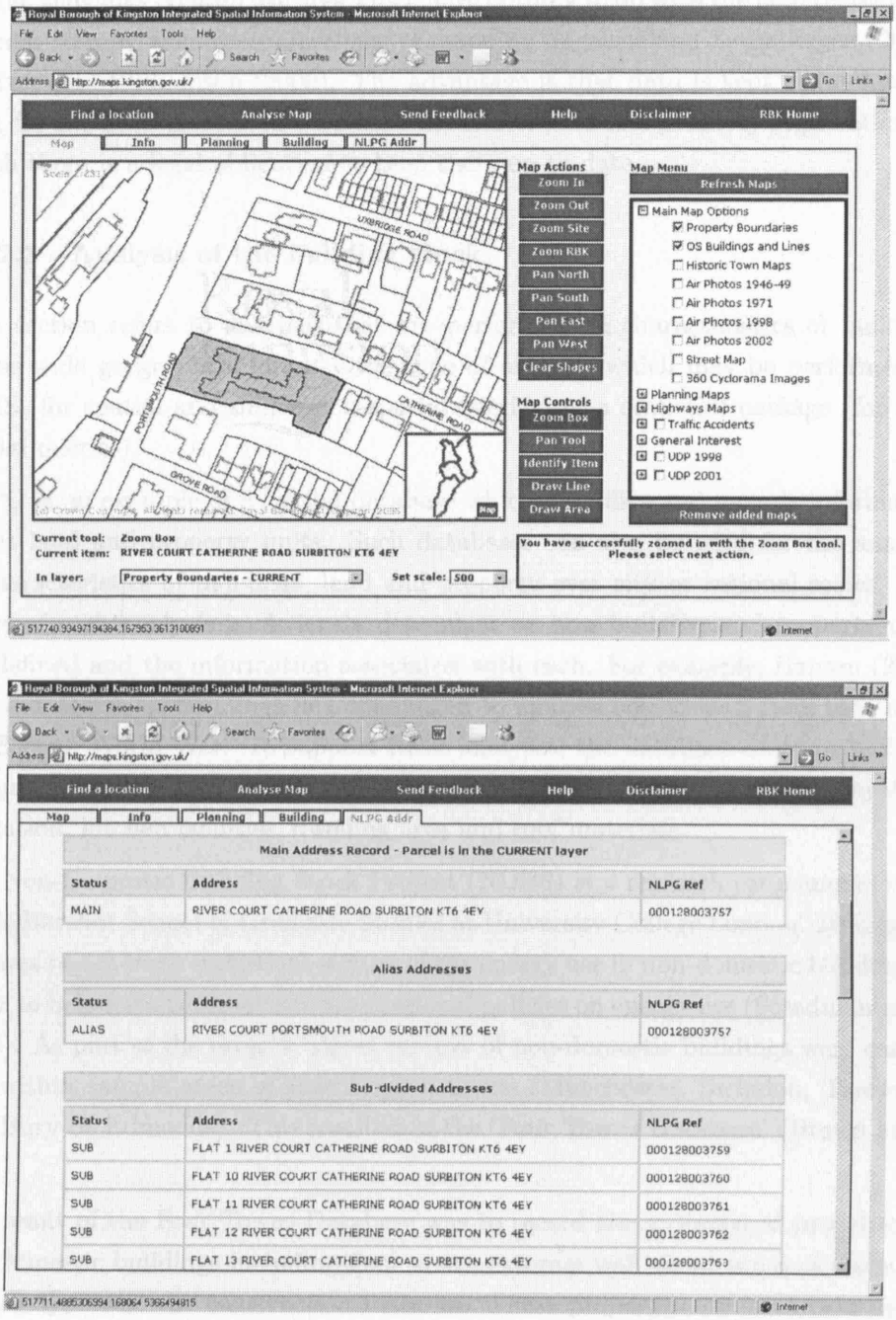


Figure 2.18: Screen capture from Kingston-upon-Thames' mapping website showing maps, addresses and planning applications for properties using data from their LLPG. The map has the property boundaries (from the LLPG) and buildings outlines (from Ordnance Survey) overlaid. A block of flats in 'Catherine Road' has been queried in this example and the extent of this BLPU is shown on the map. The NLPG address and alias address are shown below, as are the addresses of secondary BLPU units. Ordnance Survey © Crown Copyright and Royal Borough of Kingston. *Source: http://maps.kingston.gov.uk.*

However, in many countries, the land registry database is not exclusively used for land registry purposes; it also provides general purpose large-scale mapping. For this reason, they may contain features which have nothing to do with the land registration in order to provide more comprehensive mapping Bogaerts and Zevenbergen (2002); Wallace and Williamson (2006). The advantage is that data is kept up-to-date; at least for the data associated with legal aspects of land and property registration for which there is a legal obligation to keep them up-to-date.

2.9.2.3 Analysis of the building stock

This section refers to analyses that are performed on characteristics of buildings across wide geographical areas – the type of analysis which may be performed by a GIS (for spatial and simple non-spatial queries) or a statistics package (for non-spatial queries).

NLPG is an example of a spatial database which identifies and records information about land and property units. Such databases can also be used for the analysis of characteristics of buildings, land and property over city or regional scales. The range of such analysis is obviously dependent on how building and property units are defined and the information associated with each. For example, Hansen (2001) uses a database of buildings in Copenhagen to analyse city growth from the end of the Second World War. To support these analyses, the database contains building footprints with heights and attributes such as use, number of bathrooms, heating, insulation, kitchen facilities, dwelling area and roof materials.

The Non-Domestic Building Stock Project (NDBS) is a research programme based at the Bartlett School of Graduate Studies at University College London. Its original aim was to achieve a statistical picture of the energy use in non-domestic buildings in order to help form national and international policies on energy use (Steadman *et al.*, 2000). As part of the project, street surveys of non-domestic buildings were carried out within sample areas of four English towns (Manchester, Swindon, Tamworth and Bury St Edmonds). This resulted in the 'Four Towns Database' (Brown *et al.*, 2000).

The remit of the Four Towns Database was to record the geometrical properties of non-domestic buildings including floor areas, external wall areas, areas of roofs, the glazing type and the construction material. These properties were relevant to the energy efficiency estimations which formed part of the study. The surveyors broke buildings down into units such that each unit could be allocated a homogenous set of properties. The resulting units were subdivisions of individual storeys, each with a homogenous set of properties. These could then be aggregated to larger units, if required.

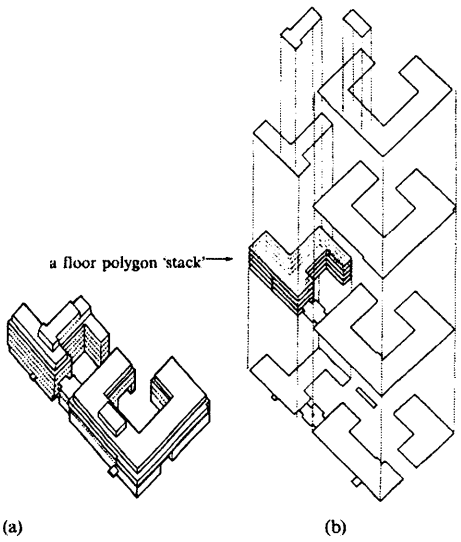


Figure 2.19: An office building, broken into floor polygons. Floor polygons can be different sizes on different storeys. *Source: Holtier et al. (2000, p5).*

The data were stored in the SmallWorld™ GIS[24]. Building units were stored as polygons representing floor areas. Each was allocated a base height (relative to the terrain) and a floor-to-ceiling height, as illustrated in figure 2.19 (Holtier *et al.*, 2000). Each polygon represents the floor area of each unit and, by extension, the unit itself. 3D output from this database is illustrated in figure 4.14.

This database was the basis for a range of analyses of the non-domestic building stock. For example, a distinction between exposed walls and shared party walls is of direct relevance to energy efficiency modelling (different implications for heat loss); thus the ratio of building volume to exposed wall area is a useful statistic. Exposed wall area can be estimated using an algorithm which tests polygon adjacency, taking into account differences in polygon height and vertical extent. Polygon edges which do not coincide with other polygon edges are taken as representing exposed wall, the area of which can be estimated. The database has also been used to study other aspects of the built environment such as providing empirical evidence for developing a classification scheme of built forms (Steadman *et al.*, 2000) and for estimating the amount of the built environment associate with different types of activity (Bruhns *et al.*, 2000). NDBS will be used as part of the new project 'CaRB' ('Carbon Reduction in Buildings', http://www.carb.org.uk/), aimed at reducing carbon emissions from the UK building stock.

The applications of the Four Towns Database described in Steadman and Rickaby (2000)[25] illustrate the wealth of analyses which would be possible for such an abstracted building database with national coverage. As typified by many of GIS-type

[24]SmallWorld™ is a trademark of the General Electric Company.

[25]Papers in the edited volume of (Steadman and Rickaby, 2000) include applications in activity patterns, building type, glazing type and area, building services classification, energy use patterns, construction type spatial patterns.

analyses, precise geometry and the high level of detail of individual CAD building models is not appropriate; more appropriate is where the built environment can be broken down into features horizontally as well as vertically. For the applications described, extruded OS MasterMap building footprints or simply the outer 3D shells of buildings are not at a high enough spatial resolution for these types of analysis.

These GIS-type spatial analyses are performed using geometrical, topological and attribute information of features. The results obtained are effectively spatial generalisations of properties of the real world. These contribute to understanding properties of the environment.

This thesis proposes that 3D national mapping should support these types of analyses.

2.9.2.4 Character of the built environment

Lynch (1960) was interested in the 'look' of cities, how their inhabitants conceptualise, interpret, orientate themselves and describe the space, and how they synthesise the character of a city from its elements. Paay and Kjeldskov (2005) develop this idea, describing MIRANDA[26] – a framework for describing urban space in terms of architectural and informational qualities. The specific application proposed was to objectively assess the characteristics of space, to find the optimal types of graphical representation for their presentation on a mobile device, in a location-based service context.

More quantified approaches are taken by others, describing cities in terms of topological aspects. Krüger (1979) develops the concept of the 'urban graph' which is used to describe the main elements of the urban environment which are classified into access characteristics (transport networks) and geometrical characteristics (ratios and adjacency). The use of graphs for describing architectural morphology is developed by Steadman (1983).

Classification schemes are another way of summarising or grouping 'things' (features, objects, words). The development of classification schemes requires the process of abstraction (as does the process of data modelling). The classification scheme of Steadman *et al.* (2000), classifies buildings according to the geometrical properties of how they are subdivided into functional spaces and whether they are naturally or artificially lit. Inferences about buildings' uses and types can be made from these (e.g. domestic and non-domestic buildings can be distinguished because domestic buildings tend to be designed so that they can be naturally lit, whereas many non-domestic buildings tend to be artificially lit and ventilated).

This section provides some examples of wider inferences that can be made from the geometrical, topological and attribute information of components of the built

[26]MIRANDA stands for 'Multilayer Information Related to Architecture aNalysis Data Abstraction' (Paay and Kjeldskov, 2005)

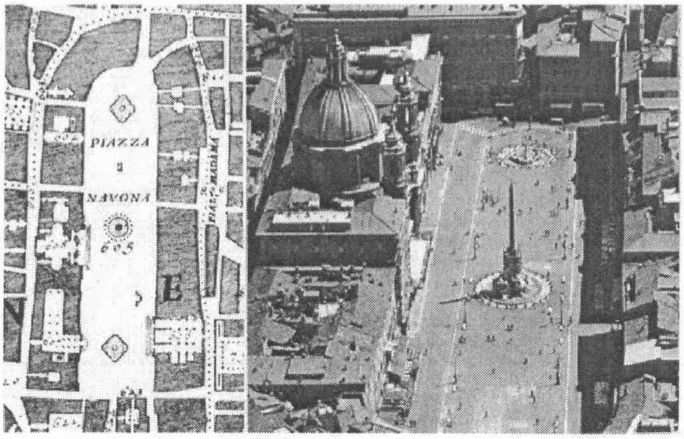


Figure 2.20: *Left:* Giambattista Nolli's map of Rome from 1748. Mediæval, Renaissance and Baroque buildings are shown in grey, Roman in black and spaces with public accessibility including the interiors of public buildings are shown in white. *Right:* Piazza Navona, Rome *Source: Hwang and Koile (2005).*

environment when analyses are made over large geographical areas, highlighting the importance of data resources for geometry, topology and feature conceptualisations.

2.9.3 Access and movement

2.9.3.1 Importance of access

Access is an important characteristic of urban areas (Liu and Zhu, 2004; Lee, 2004b; Talen, 2002). As early as 1748, Giambattista Nolli's map of Rome showed it as a mosaic of public and private spaces (figure 2.20), the boundaries of which were defined not only by streets and parks, but also by privately owned interior public spaces. Such approaches have been useful for urban design and study (Hwang and Koile, 2005).

In the 1960s, with the rise of personal motorised transport, radical new ideas for the restructuring of cities were proposed in which areas of human activity and pedestrian movement were segregated from a highly efficient road infrastructure. Marshall (2005), in his book on street patterns, begins with a stark illustration of the possible transformation of the Bloomsbury area in London if it had been redeveloped based entirely on such ideas (a transformation at this extreme is unlikely to have ever been carried out). Characteristics based on toned down versions of these ideas are evident in some of the architecture of the period (such as those illustrated in figure 3.2) where pedestrians are taken off roads in 'conduits' which integrate with large built complexes in which cars have no place. This leads to a complicated intertwining of geometry and access which is often difficult to navigate and is often characterised by an abundance of wall-mounted maps.

In the past couple of decades, there has been a renewed interest in walking, cycling and public transport, and in encouraging street life and compact urban areas with

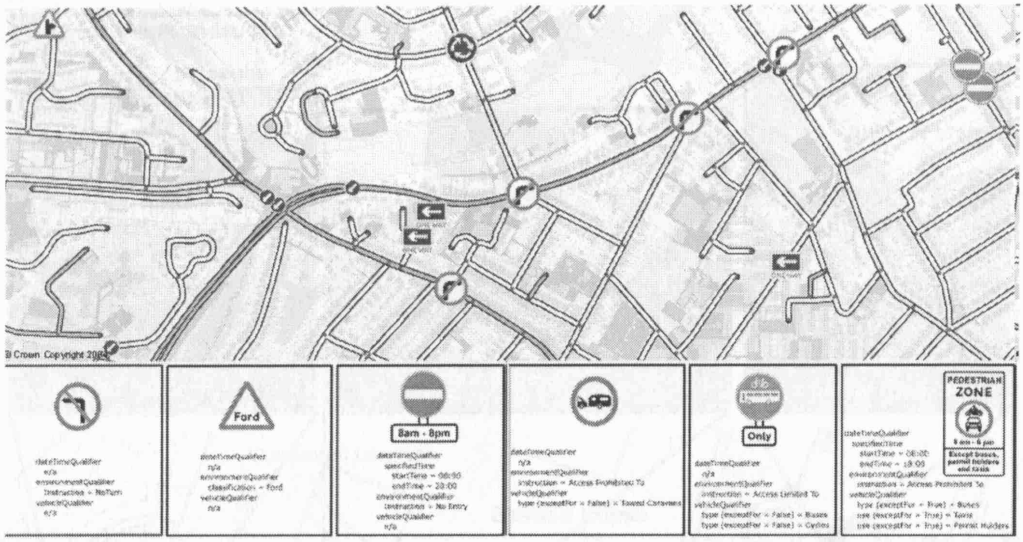


Figure 2.21: Illustration of some of the 'Road Routing Information (RRI)' embedded in ITN. The specific examples show that each RRI has a 'dateTimeQualifier' (the time or period for which the RRI is valid), an 'environmentQualifier' (the nature of the RRI) and a 'vehicleQualifier' (to whom the RRI applies). *Source: http://www.ordnancesurvey.co.uk/oswebsite/products/osmastermap/demos/itn_tech/pages/rri.html*

mixed land use (Marshall, 2005). Talen (2002) suggests that for this reason, ease of walking and general accessibility should be a factor in planning, something which is not currently the case. Marshall (2005) devises a range of urban-wide quantitative measures of different aspects of the forms street patterns, in the hope that these may be used to help identify good practice in design.

Access in built environments is generally optimised to certain agents of access and may be completely unsuitable for other agents of access. Motorway intersections with motorway service stations are optimised to fast-moving motorised vehicles. Pedestrians are only permitted in certain areas of the service stations. Pedestrian zones in city centres are unsuitable for most motorised transport. Outside the pedestrian zones in city centres, it might be that access by certain vehicles is penalised; this is the case in any city in which roads have separate lanes for public transport. The congestion charge in London is a more extreme example.

2.9.3.2 Describing access

Access can be conceptualised as topological relationships between discrete spaces. This can be described using a graph, as explained in section 4.7.1. Attributes of the topological relationship describe characteristics of access.

The 'Integrated Transport Network (ITN)' is a layer within OS MasterMap which is used to describe vehicular access. Further information about access is given as attributes of the nodes and links of the graph. These are termed 'road routing information (RRI)' and are illustrated in figure 2.21. Some of these are specific to particular types of vehicle and particular times of day.

Figure 2.22: An illustration of the complicated pedestrian access issues around some building complexes. An extract from Land-Line for the Museum of London is illustrated with photographs showing the complicated pedestrian access arrangements over multiple storeys. Most maps do not describe the pedestrian access very well. *The map is Crown Copyright Ordnance Survey. An EDINA Digimap/JISC supplied service*

The 'Integrated Transport Network' is an example of a *geometrical network*, because the links not only represent topological relationships linking spaces but also serve to represent the (2D) geometry of the link. These correspond to road centrelines.

When dealing with road networks, usually it is only 'official' routes that are included, i.e. the routes provided by national or regional governments. Such routes provide a fairly complete picture of vehicular access. However, when dealing with pedestrian access at the microscale (i.e. at the scale of the individual) especially when considering the interiors of buildings, the complete set of 'official' routes of pavements and footbridges represents only a subset of where a pedestrian is likely to walk. Thus, spaces to which pedestrians have access are more diverse in character than roads are for vehicles. The majority of well-maintained roads are suitable for the majority of road vehicles. The same is not true to the same degree for pedestrians because there is a wider variation in the routes which are in use by pedestrians. Also, vehicles tend to be restricted to road networks, whereas pedestrians are not restricted to such networks; they move through open spaces.

Most large-scale maps are suitable for pedestrian navigation, but access to buildings is often poorly depicted. This is problematic in terms of pedestrian navigation, es-

Figure 2.23: Map of the Museum of London, intended to show pedestrian and wheelchair access. A symbol (an 'A' in a circle) is used to indicate where there is access to a elevated walkway to the museum ('highwalk'). *Source: http://www.museumoflondon.org.uk/English/MOL+MAP.htm*

pecially where there are large buildings or buildings with open public routes passing through. Figure 2.22 shows a map of the front of the Museum of London. The map is not designed to show pedestrian access; it does not even attempt to. The photographs show the great complexity of pedestrian access issues spread over several layers. Figure 2.23 is a map provided by the museum itself and is intended to show pedestrian access to the site. It does this by providing symbols indicating where access to the elevated walkways can be gained.

Figure 2.24 shows a map of the main campus site of University College London. Access to buildings is shown with the use of arrow-heads whose colouring indicates whether access is suitable for wheelchair users. Main routes through buildings are depicted by dotted lines.

Floor plans to aid navigation are common occurrences in large buildings, particularly to which there is public access. The Barbican (London, UK) is a cultural and residential development with complicated multi-level pedestrian accessibility issues. Figure 2.25 shows a 3D interactive map for The Barbican (London, UK) from its website, designed to try assist users' conceptualisation of the building. It shows a perspective view of the storeys and allows them to be scrolled and allows conventional floorplans to be viewed separately. Such maps are for human interpretation which aim to assist pedestrians in navigating, rather that be used for computer-based interpretation to support automated routing assistance or emergency planning applications. A solution proposed by Ordnance Survey is a version of ITN with connections to the access points of buildings. An example of some test data can be

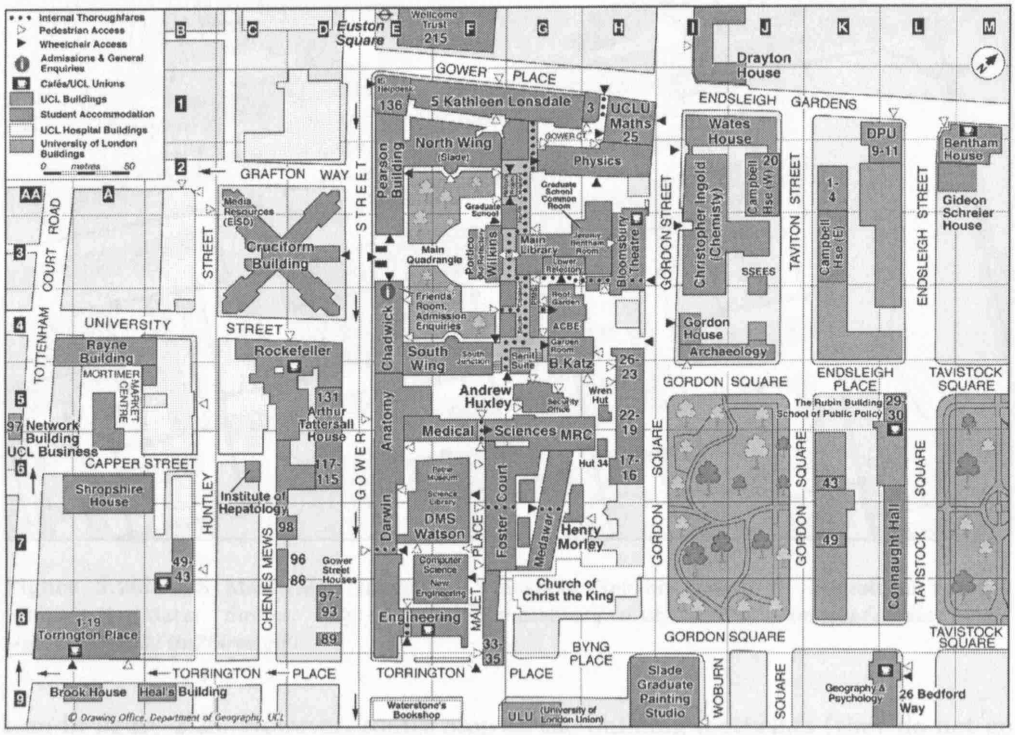


Figure 2.24: Pedestrian map of the main campus of University College London. Triangular arrowheads are used to indicate access including access suitable for wheelchairs and dotted routes indicate thoroughfares through buildings. *Source: http://www.ucl.ac.uk/about-ucl/images/image_bank/maps/main_site_colour_2005*

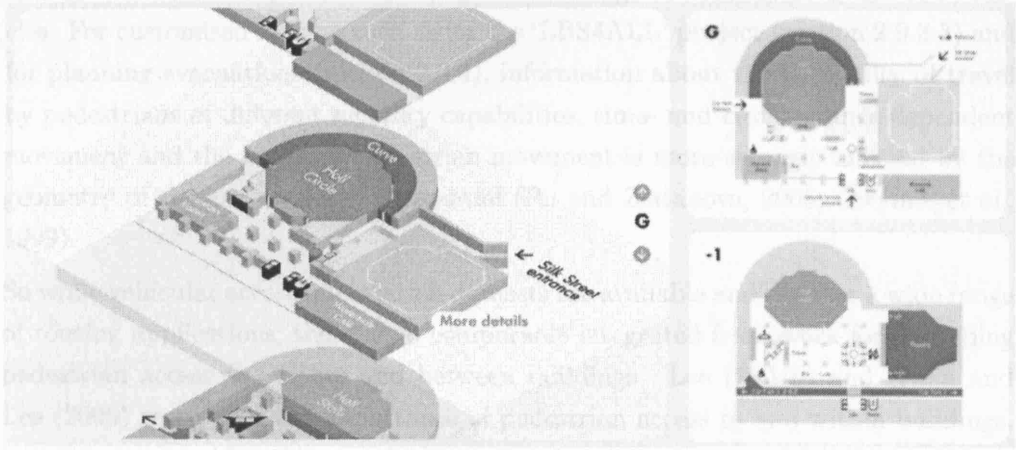


Figure 2.25: Interactive map of The Barbican, London. Access to and within The Barbican is shown by the interactive 3D map available on their website. Using the up and down arrows, the user can scroll up and down the storeys. Each storey has a 2D planometric maps, two of which are shown on the right. *Source: http://www.barbican.org.uk/visitor-information/barbican-venue-maps*

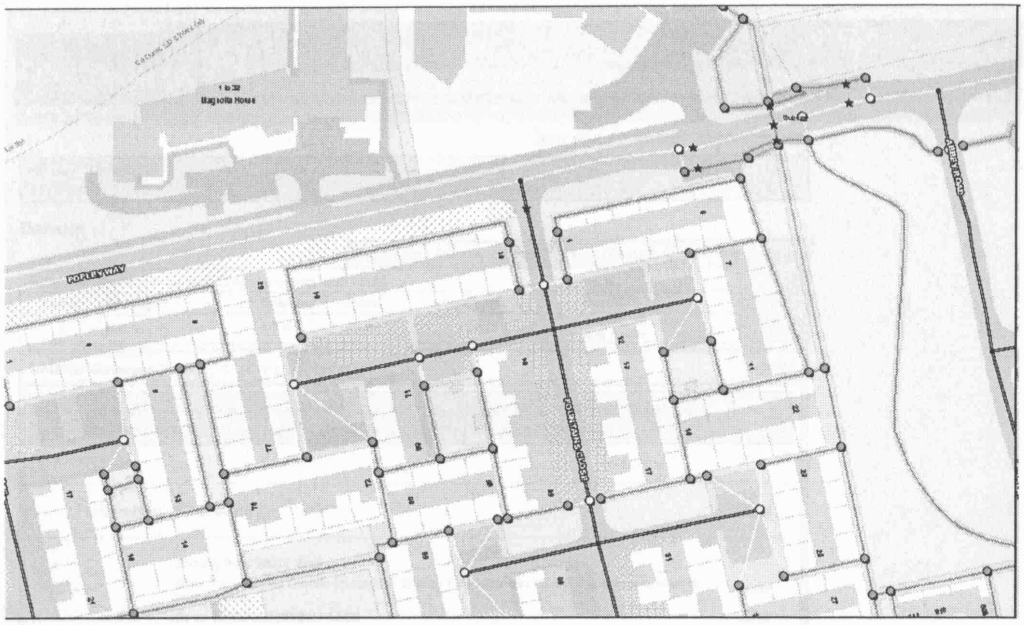


Figure 2.26: OS MasterMap Integrated Transport Network phase 2: Pedestrian network – illustrative data. *Source: http://www.ordnancesurvey.co.uk/oswebsite/images/media/news/september2003/itn2hires.gif.*

seen in figure 2.26. However, routes stop at the building footprints (they do not go inside), the information does not encapsulate the complexities of pedestrian access, and it has never been released as a complete product. Other than the test data, no national dataset of pedestrian access exists.

The complexities of access for pedestrians are more complicated than those for vehicles. This is because everyday pedestrian access is over and through a wider range of environments than vehicular access, which is restricted to roads in an everyday context. In addition, pedestrians have a wider range of abilities than do road vehicles. For customised routing such as for the 'LBS4ALL' project (section 2.9.3.3) and for planning evacuations (section 2.9.4), information about the suitability of travel by pedestrians of different mobility capabilities, time- and circumstance-dependent movement and the fact that pedestrian movement is more strongly affected by the geometry of the environment, is essential (Pu and Zlatanova, 2005; Gwynne *et al.*, 1999).

So while vehicular access information datasets are available and support a wide range of routing applications, there is no comparable integrated framework for describing pedestrian access to, within and between buildings. Lee (2004b) and Kwan and Lee (2005) recognised the importance of pedestrian access to and within buildings, but did not consider the complexities of pedestrian access and do not deal with access through open spaces; access is restricted to routes constrained to centrelines of corridors generated by their S-MAT ('straight medial axis transform') algorithm.

Figure 2.27: Example of a web-based road routing service. The Transport Direct website suggests a route between a specified starting point and destination at a specified time on a specified date. This figure shows the first part of the suggested route for a car starting from the Centre for Advanced Spatial Analysis at University College London (WC1E 7HB) travelling to the Department of Geography at the University of Edinburgh (EH8 9XP). The London congestion charge is shown because it applies at the time and date of travel. *Source: http://www.transportdirect.info/*

2.9.3.3 Applications

Figure 2.27 shows an example of part of a road route between two places from the Transport Direct[27] website. Such routing applications are increasingly using more up-to-date information on traffic volumes, road closures and accidents. Vehicle satellite navigation systems allow these data and information about vehicle's location, to be incorporated, making for a system which is reactive to current location and uses up-to-date road information. Such systems are in increasingly common use and are changing the way in which drivers navigate and behave. For example, a recent BBC news item[28] blamed satellite navigation systems for causing a large increase in traffic in a small Somerset village as a shorter alternative to using the main road.

[27] http://www.transportdirect.info/

[28] http://news.bbc.co.uk/1/hi/england/gloucestershire/4781350.stm

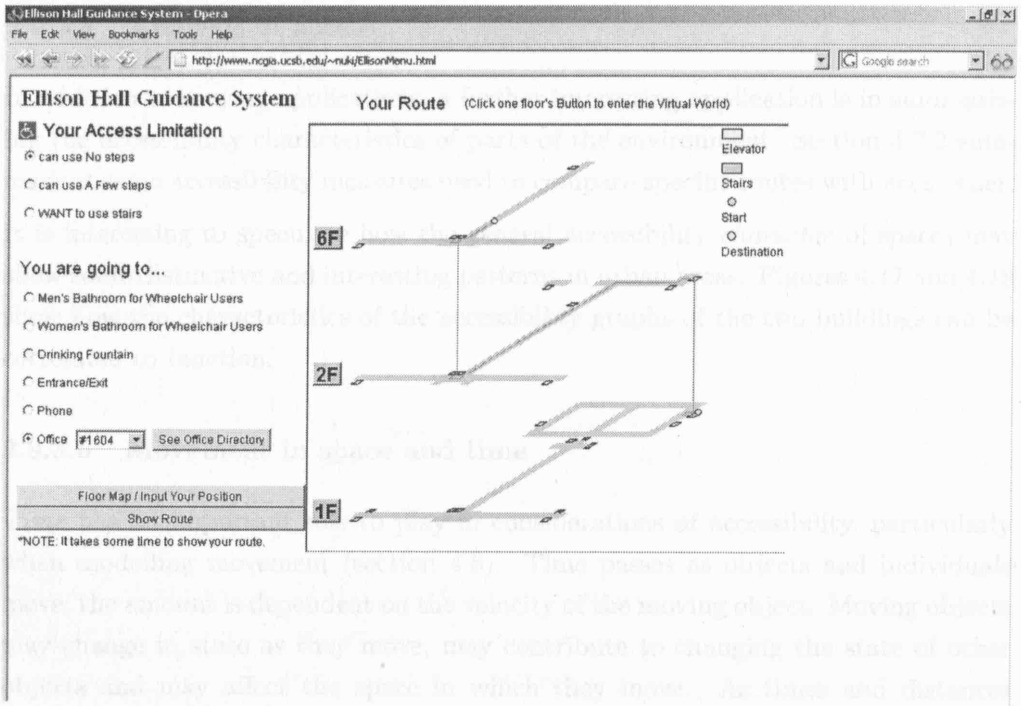


Figure 2.28: Java applet which finds a route within a building for pedestrians with access limitations. *Source: http://www.ncgia.ucsb.edu/~nuki/EllisonMenu.html*

Another recent article[29] describes how satellite navigation systems are being used by drivers to avoid being caught on speed cameras and to plan shopping trips.

In terms of pedestrian movement, the 'LBS4ALL'[30] project is investigating the delivery of location-specific information on mobile devices for people with mobility problems. The research project looks at how relevant information can be provided to the user when needed. Okunuki *et al.* (1999) built a simple web-based system to help guide people – particularly those in wheelchairs – to chosen destinations in buildings[31]. Screenshots of the applet are shown in figure 2.28. This is a very simple example with very simple restrictions on pedestrian movement.

For applications of large-scale agent-based pedestrian modelling (modelling the behaviour of pedestrians as individuals rather than flows), such information is important. For some applications, the seamless representation of access inside and outside of buildings is desirable. One such application is that of emergency evacuation in section 2.9.4 in which the heterogeneity in pedestrian characteristics may cause problems for crowds leaving through narrow exits.

[29] http://news.bbc.co.uk/1/hi/magazine/4124760.stm
[30] http://lbs4all.soi.city.ac.uk/
[31] http://www.ncgia.ucsb.edu/~nuki/EllisonRep.html

2.9.3.4 Analysis of access

In addition to routing applications, a further interesting application is in summarising the accessibility characteristics of parts of the environment. Section 4.7.2 summarises some accessibility measures used to compare specific routes with each other.

It is interesting to speculate how the general accessibility *character* of spaces may show some distinctive and interesting patterns in urban areas. Figures 4.17 and 4.18 show how the characteristics of the accessibility graphs of the two buildings can be correlated to function.

2.9.3.5 Movement in space and time

Time has an important role to play in considerations of accessibility, particularly when modelling movement (section 4.8). Time passes as objects and individuals move; the amount is dependent on the velocity of the moving object. Moving objects may change in state as they move, may contribute to changing the state of other objects and may affect the space in which they move. As times and distances become larger, as the numbers of interacting objects increase and as conceptual models become closer to real world representation, so the static view of the world becomes an increasingly inadequate simplification of a dynamic world in which the state changes through time.

Routing and timetabling applications need to consider the departure and arrival times of services and to allow time to change between services. Figure 2.27 shows the role of time in calculating routes, with details such as whether the congestion applies at the projected time. Another example is the Transport for London Journey Planner[32] which advises on travel on public transport within London.

When dealing with events in sequence, there is the possibility of feedback cycles in which an event causes a change in the state which affects subsequent events.

Some maps depict a temporal component; e.g. some public transport maps have route lengths proportional to average travel time rather than distance. A travel route on a map may have varying thickness to indicate speed or varying colour to indicate time. However, the most influential and successful formal model of movement in space and time was developed by Torsten Hägerstrand in 1970. He saw time and space as inseparable and conceptualised them as combined in a space-time cube, in which the x and y axes represent 2D Euclidean space and the z axis represents time (they must be three dimensional – 2D Euclidean space and time) . Hägerstrand's space-time cubes are an enormously useful generic framework for conceptualising the relationship between space and time. Kraak (2003) demonstrates their usefulness for visualising spatio-temporal data.

[32] http://journeyplanner.tfl.gov.uk/

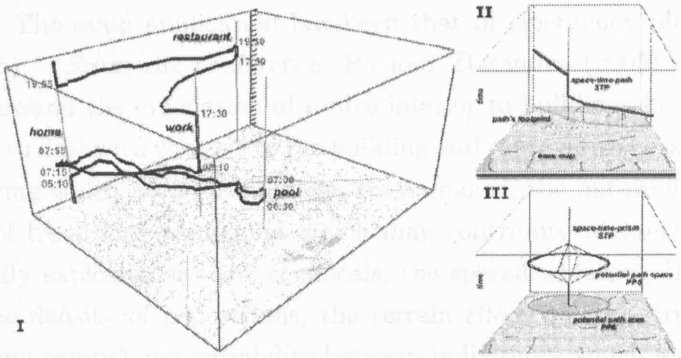


Figure 2.29: The space-time cube. *(I)* An example of a pedestrian's movement in time and space plotted in a space-time cube. *(II)* A space-time path and its footprint. The vertical line represents a location over time. *(III)* A Space-Time-Prism (STP) indicates the locations that can be reached in a particular time interval. The projection of this onto the map (the Potential Path Space (PPS)) represents the Potential Path Area (PPA). *Source: Kraak (2003, p1989).*

Figure 2.29 illustrates concepts within the space-time cube. As an object moves in 2D Euclidean space, it correspondingly moves in a horizontal plane through the space-time cube. As time passes, the horizontal plane representing space moves vertically along the z axis. The resulting space-time path can be plotted in 3D space. As with any plot in 3D space, any one dimension can be excluded by projecting the plot onto a plane perpendicular to the axis of the dimension to be excluded – thus the footprint of a space-time path represents the path in 2D space without time. A space-time prism (STP) is a volume within the space-time cube which represents all the space accessible within a time interval (depending on the velocity at which the object moves).

Space-time cubes have inspired a large number of studies on the interaction of individuals with space and time; for example, Kwan (2002) looks at the spatial and temporal arrangement of human activities.

2.9.4 Emergency planning

A recent conference in Delft (the Netherlands) on the use of geographical information to support various stages of disaster management, had a range of papers on case studies, requirements, data collection, user interfaces, positioning and information systems for this purpose (van Oosterom *et al.*, 2005). This highlights not only the substantial interest in this area (Goodchild, 2006) but also that the effective combination of diverse strands of planning, geography and information technology are required to support planning for emergencies. Of the issues already raised, geolocation, data collection, representation (interiors and exteriors) and access (to interiors and exteriors) are highly relevant.

Lee (2004b) recognises the importance of considering space, at the microscale, both exterior and interior to buildings, access between these spaces, other topological relationships (Lee and Kwan, 2005) and georeferencing within buildings

(Lee, 2004a). The main application has been that of emergency planning (Kwan and Lee, 2005). From the conference, Pu and Zlatanova (2005) consider some of the issues around the evacuation of routes interior to buildings in an emergency, including how to deal with changes to the building and route during evacuation. This includes knowing which parts of buildings are damaged, the nature of the damage, the location of hazardous equipment which may contribute to the aggravation of fires, potentially explosive or toxic chemicals, the spread of fumes, the capacity of the routes, the density of pedestrians, the terrain effect on pedestrian movement speed (steps and ramps) and variability between individuals which may affect their movement and the movement of others (Pu and Zlatanova, 2005; Gwynne *et al.*, 1999).

To support these types of application, a model of the state of the building in its environment is required. In Britain, organisations are responsible for ensuring that their buildings comply with various fire safety regulations and pass building plans annotated with information about hazards (e.g. locations of the storage of flammable or explosive chemicals or hazardous equipment such as deep fat fryers), access points and escape route information to the local fire service. Examples in the following two paragraphs show how these data are being used and ongoing research.

The CAD centre at the Greater Manchester Fire Service in Bolton[33] holds floorplans of buildings, mostly in DXF format. As discussed in section 2.6.1, these types of files are relatively unstructured and these 'digital drawings' are held for human interpretation.

The Royal Berkshire Fire and Rescue service[34] has been experimenting with using GIS systems as part of 'Electronic Service Delivery' project, driven by the feature-based nature of OS MasterMap. This project investigated the use of a spatial database whose base data was OS MasterMap, where the TOID was used as a 'hook' or placeholder for recording information. For example, individual call-out incidents were referenced to individual TOIDs. These data were used to identify patterns in call-outs, targeting properties where false alarms were unusually high, and encouraging steps to be taken to reduce this number.

Integrated emergency planning using GIS and other similar systems is very topical. It is also highly demanding, requiring up-to-date and detailed data, delivery systems which can provide the information where and when needed, and managing situations which are dynamic in nature. It highlights the importance of access information (access exterior and interior to buildings and different types of access) and the importance of georeferencing (for the geolocation of hazards and individuals).

[33]Personal communication: Tom Wilkes, senior CAD technician, CAD centre, Greater Manchester Fire Service.

[34]Personal communication: Jonathan Ball, Royal Berkshire Fire and Rescue service.

2.10 Purpose of digital base mapping

2.10.1 Geometry

The purpose of base mapping is to provide a geometrical representation of ground-based detail whether this be of features with a physical presence (section 2.3.2) or non-physical features (section 2.3.3). As discussed, in a digital context, holding these geometries as distinct features in a common framework is an advantage. Thus far, base mapping has provided 2D geometry inherited from its paper-mapping origins. Since the tools for which it is used are in two dimensions (GIS) and because 3D data of this type is difficult to obtain and 2D data is adequate for the majority of end-users, mapping has remained two-dimensional.

There is increasing interest in the provision of 3D data. This is well established in architecture and engineering using CAD software tools. For the scale and coverage of GIS-type tasks for which national mapping data is aimed, these are too detailed and have an undue emphasis on geometry. A large range of applications reviewed in section 2.9 show the importance of a data resource which allows the built environment to be analysed and summarised, in order for it to be better understood. These applications require information about buildings which is more detailed than in current large-scale national mapping, yet less detailed than individual building models, and in a framework which allows them to be compared over large geometrical areas. Frank (1994a) shows that the identification of features and spatial relationships between features are often more important than the precise geometrical character of the features themselves. Section 2.4.2 also shows Ordnance Survey's practice of inferring an internal boundary of a building where it is known that a portion fulfills a significant role; i.e. the fact that this portion exists is more important than its precise boundary delineation.

Ordnance Survey's DNF® aims to provide a framework for all providers of base mapping to offer their mapping in a common framework and all users of geospatial data to have the means to combine this with their own data. The framework incorporates a 'feature-based' approach with an inbuilt discrete georeferencing system identified by 'TOIDs'. A feature identified by a 'TOID' thus provides a 'hook' onto which information and identity can be attached.

Many of the applications reviewed in section 2.9 deal with units of buildings which have a 3D extent and can be geolocated in 3D, and it is clear that it is appropriate to be able to describe the 3D geometry of these. The applications also show that the structure of space in which an activity can take place is important. The common practice of providing a description of the outer shells of buildings in this context is inadequate for most of these applications. These are the types of applications at which 3D national mapping ought to be aimed.

2.10.2 Features

Traditional mapping usually classifies features using colour, symbology and a key. The features being classified are defined by the data provider. This concept is usually extended to digital base mapping. Ordnance Survey provides features in its OS MasterMap product. The current model of use of OS MasterMap is to use features as supplied through the TOID.

The problem with base-mapping is that although it provides geometry (topography), because there are no natural units of space, geometry must be provided on some conceptualisation of features. The reviewed applications all deal with different units of the built environment whose defining criteria differ. Their direct comparison in analysis is usually not straightforward.

The ability of base mapping to provide geometry for many different features which happen to nest and overlap in a complicated manner would increase the applicability of the base mapping to support a much wider range of spatial databases of the environment.

2.10.3 Access and topology

Access is clearly important in the built environment. The specific application of emergency planning (section 2.9.4) shows the importance of access in the context of the geometry and other aspects of the built environment. Many of the other applications would also benefit from a description of accessibility.

2.11 Summary

This chapter has shown the importance of models and explicit frameworks for representing the real world to allow their use in a wide variety of applications. It has shown how digital topographic base mapping, when based on an explicit framework (using OS MasterMap and DNF as examples) is an important resource for mapping data. The subsequent review of applications has shown that current digital national mapping, particularly when it does not capture the structure of the built environment, is inadequate for a wide range of applications in which information on buildings over large geographical areas is of interest. The types of application identified as relevant are those which rely on spatial databases of the real world for GIS-type applications. Although CAD has been reviewed, the emphasis it places on the precise and detailed geometry of individual buildings has been shown to be inappropriate.

Section 2.3.6's reflection on 3D digital mapping and the applications reviewed in section 2.9 identified that more interesting than the external geometrical shell of buildings (as is often modelled in virtual cities, section 2.9.1) is the *internal structure of*

functional spaces. Section 2.9.2 identified applications which need to analyse or summarise characteristics of the built environment over large geographical scales. The topology between these spaces is essential for many GIS-type analyses. The topology of pedestrian access is also important, as is a rich description of pedestrian- and context-dependent constraints to access. Applications for accessibility information were reviewed in section 2.9.3 and were seen to be very important for the application domain of emergency planning (section 2.9.4).

The lack of any natural units of space means that any description of geometry must have spatial units defined. These are usually conceptualisations of features. However, as illustrated for land registries, gazetteers and building stock analysis, conceptualisations of features for topographic base mapping are not necessarily compatible with other organisations' conceptualisations.

It has also been shown that time adds an important dimension. Events and history are important factors in the identity of features, causing them to change in state, and potentially causing the spawning of new features. In terms of access, time is incredibly important – it is a major constraint to access. Over the time in which access is being gained, the constraints may change.

This chapter identifies the need for a feature-based database for large-scale national mapping, which incorporates 3D geometry, multiple feature conceptualisations and pedestrian access. The types of applications this should support are GIS-type applications (i.e. concerned with analysis and comparison over large geographical areas rather that concerned with the details of individual buildings) and the examples of the applications to support are listed in section 2.9.

Chapter 3

Design issues

The previous chapter identified the need for a feature-based mapping framework which is capable of storing the structure of spaces in the environment, representing pedestrian access between these, providing a facility to identify real-world features and a description of time. The emphasis is on the built and urban environments, but the same principles can apply to the natural environment. A set of target GIS-type applications were identified in section 2.9. These were applications which rely on inventories of building units over large geographical areas, pedestrian movement applications and analyses, emergency planning and, to a lesser degree, 'virtual city' models.

In the light of the previous chapter, this chapter develops and presents a set of design requirements and principles to guide the design of a framework for a model which can support applications such as these. These design requirements are summarised below:

- the ability to describe different conceptualisations of features
- the seamless treatment of space exterior and interior to buildings
- pedestrian accessibility
- the concept of a data repository to which information can be added in an incremental fashion
- the storage of 2D and 3D geometry
- the representation of time

In chapter 2 (particularly section 2.3.6), it was seen that it is the *structure of functional spaces* in the built environment which is of interest. This sets the aims of this framework apart from many of the geometry-centred efforts of the examples of virtual cities (section 2.9.1) and CAD models (section 2.6.1). Compared to these geometrical descriptions, the one used by this framework will be rather simpler, because it is the extent of functional spaces which is of interest, rather than precise 3D geometries; for example, roofs are out of scope.

3.1 Conceptualisations of real-world features

All large scale vector topographic maps partition space according to feature boundaries. These partitions of space correspond to conceptualisations of real-world features in some manner (e.g. figure 2.3). Each feature has an identity and its conceptualisation involves identifying which aspects of it are key to its identity. Features in large-scale topographic mapping are conceptualised by the data provider. Aspects of a feature include its classification, its geometry, its state (attributes) and its history. This thesis does not address the *process* of conceptualising features (more detail about this can be found in section 4.6). Instead, it assumes that a feature has been conceptualised, by whatever means, and the feature has an identity which can be given a unique identifier.

An OS MasterMap feature is based on a point, a line or a polygon geometry, and has a unique identifier, known as a TOID. Some of these geometries – particularly the polygon geometries – correspond to real-world features, examples of which are fields, parks and buildings. The conceptualisation of features is governed by OS MasterMap's capture specifications (Ordnance Survey, 1996).

Section 2.4 discussed boundary delineation. Uncertainty and fuzziness in boundary delineation is predominant in natural environment features. Although urban boundaries tend to be sharply defined, the ways in which the geometrical extent of features can be subdivided are diverse, and this affects feature definition. As comprehensively reviewed in the previous chapter, there is a seemingly infinite potential range of conceptualisations of features, and the case studies in section 2.9.2 show that different applications require different feature conceptualisations. Particularly for buildings, the section shows that analyses across large geographical areas require the built environment to be broken up and aggregated in different ways. Section 2.4.2 shows Ordnance Survey's definition of buildings, but section 4.6.1 reviews other criteria which can be used to define buildings. The 'Four Towns Database' of the Non-Domestic Building Stock (NDBS) project decomposed building into units based on each having a unique set of surveyed attributes (section 2.9.2.3). There is a mismatch between NLPG BLPUs, OS MasterMap TOIDs and Royal Mail's delivery points (I&DeA, 2004).

A range of properties which may be used for defining the extent and classification of buildings have been identified in section 4.6.1. These include physical, historical, functional, legal/conventional and accessibility properties. If the framework supports the storage of such properties, it can support the automatic definition and classification of buildings. A similar approach can be used to identify, delineate (section 4.6.2) and classify features (section 4.6.4).

When dealing with subdivisions of buildings, building interiors are also dealt with; this is relevant for the seamless exterior/interior design principle (section 3.2). Since

these subdivisions exist on multiple storeys, this is also relevant to the 3D representation design principle (section 3.5).

3.2 Towards the seamless treatment of space, exterior and interior to buildings

When considering 3D national mapping, there is a common emphasis on the 3D external shells of structures. This is typified by many of the examples of virtual cities (section 2.9.1). In a national mapping context, it could be argued that this is reasonable, because national mapping agencies are generally interested in the external geometrical forms of structures, as shown in the previous section. However, as shown in Ordnance Survey's specifications in section 2.4.2 and figure 2.11, this includes the functional units of terraced housing which can be fairly reliably inferred from clues external to the terrace structure (but not always, as shown in figure 2.12). In a 3D context, the identification of spaces on multiple storeys becomes possible and an extension to the approach of inferring internal boundaries from the exterior could be to make inferences from the size and position of windows on different storeys. As argued in the previous chapter, in a GIS-type context the identity and topology of features is often more important than the precise geometry. Some of the applications on city-wide scales reviewed in chapter 2, namely land and property registration (section 2.9.2.2) and building stock applications (section 2.9.2.3) require such information.

The examples shown in figure 2.16 usually have the 3D forms modelled as hollow external shells. This does not add much new functionality to the model (except visual effects and viewshed-based analysis of the external envelope of the built environment). If, rather than being hollow, the buildings had their interiors modelled, this would add a great deal more functionality (although this is more difficult in terms of data acquisition), because it would be possible to identify spaces *within* buildings. Figure 2.1 shows an old Ordnance Survey map depicting a building interior, while a couple of centuries earlier, Giambattista Nolli selectively included some interior spaces in his map of Rome (figure 2.20).

Before going any further, it is worth exploring a philosophical point. 'Inside' and 'outside' depend on the concept of a something to be inside or outside of. It thus follows that the concept of space 'exterior to a building' and space 'interior to a building' is dependent on a particular conceptualisation of a building. As shown in the previous chapter, although there may be a broad agreement on what a building is (e.g. 'a roofed construction'), for the purpose of being inside of it or outside of it, there is no unambiguous and rigorous definition. Figure 3.1 shows a shopping centre, in which much of the space outside the shops is open to the elements. This blurs the distinction between what is 'inside' and what is 'outside'. If 'inside' and 'outside' were modelled in different systems how to partition the world into 'inside'

Figure 3.1: Views of a small shopping centre showing a blurred distinction between 'exterior' and 'interior' space. The galleries on the different levels are mostly uncovered. The escalator (upper right) is in a covered, but not walled area. Most of the galleries are completely or partially covered (upper and lower left). There is pedestrian access through the shopping mall (lower right) and all the shops form distinct units.

(to be stored in one system) and 'outside' (to be stored in another system). This dependency on the conceptualisation of buildings can be removed by treating all space seamlessly. 'Inside' and 'outside' spaces could then be conceptualised using rule sets.

There are also more practical reasons why spaces should be treated as seamless. In a retail modelling application, it might be appropriate to treat all functional retail units with equivalence ('inside' a shopping centre 'building' or on the high street). This would be difficult if 'outside' and 'inside' for individual shopping centre 'buildings' were modelled in different systems. A pedestrian modelling application for the same application might wish to treat pavements, uncovered pedestrian areas, uncovered walkways (figure 3.2) and corridors within an indoor shopping centre with equivalence. As observed in section 2.9.3, some of the modernist design principles of the 1960s and 1970s led to building complexes with complicated and intertwined arrangements of pedestrian conduits (Chang, 2002). The seamless connection of pedestrian accessibility both inside and outside buildings is potentially very useful. The London Transport Demonstrator prototype (p68) experimented with including the interiors of some buildings for navigation, as has Lee (2004b).

In summary, the concept of 'inside' and 'outside' is often ambiguous, is often not useful, and most importantly, is tied to predefined 'buildings' (to delineate 'inside' and 'outside' space) of which there is no universal definition.

3.3 Pedestrian accessibility

Access is a tremendously important and interesting aspect of the urban environment. Section 2.9.3.2 reviewed approaches to supplying data on access. It identified that although a national dataset for vehicular access exists in widespread use, the same is not true for pedestrian access. Pedestrians have access to *and* within buildings. Also, since pedestrians move over and through environments with a wider range of characteristics and because pedestrians have a wider range of abilities and speeds, the description of pedestrian access is much more complicated than it is for vehicular access. This has been identified as important for emergency evacuation (Pu and Zlatanova, 2005; Gwynne *et al.*, 1999).

Related to the 'seamless space' design principle, figure 3.2 shows photographs of the complexities of pedestrian access in and around elevated walkways. Navigation is not intuitive and pedestrians are greeted by a large map in which the structure is abstracted to 2D (shown in figure 3.3) which, in a similar manner to the map for the Museum of London, shows access points and the 'highwalks' (elevated walkways) are marked.

Figure 3.2: Photographs of the Barbican (London, UK), showing complicated multi-storey pedestrian accessibility issues similar to those for the Museum of London (figure 2.22). The Barbican Underground station (left) has direct access to the (open top) elevated walkway which crosses the road and leads straight to the Barbican site. The image on the right shows more elevated walkways. The third image from the left shows that some of the walkways are covered.

Figure 3.3: Pedestrian map of The Barbican (London, UK), showing access points to the Barbican marked by numbers. Most of the access points are via elevated walkways.

3.3.1 The need for framework support of pedestrian access

As argued in section 2.1, digital models should be underpinned by a strong and explicit framework, in order to facilitate computer-based interpretation for restructuring the data where required for analysis. For the reasons discussed, just as 'inside' and 'outside' space is to be treated seamlessly, so too 'inside' and 'outside' pedestrian accessibility should be treated seamlessly. Sakkas and Pérez (2005) note that when assessing accessibility within buildings it is usually important that access can be gained to the building site from bus stops or car parks; thus it is necessary to consider seamlessly the space exterior to the building. The reason for this is that even if a building is very easy for a wheelchair user to move around in, if the user cannot enter the building because of a steep flight of steps, for example, a false impression is given of the success of the building for facilitating access to wheelchair users.

Since pedestrian accessibility is such an important part of the built environment, a framework which embeds a description of pedestrian accessibility within the geometry of the built environment will support a whole range of new applications, including those involving pedestrian modelling and location-based services. It is also possible

that interesting properties of the built environment can be inferred from the pattern of pedestrian accessibility. For example, the pedestrian accessibility patterns for sites such as the Barbican are distinctly different from traditional residential or cultural areas.

3.3.2 Factors affecting pedestrian access

Microscale pedestrian access is potentially more complicated to describe than the common approach of describing vehicular access by fully-connected networks with attributes on the network elements.

The geometry of the built environment affects access: the dimensions of corridors, doorways and rooms, the gradients of slopes and the heights of steps. The topological connectivity of spaces by walkways, ramps, steps and lifts is also essential. The geometry and the topological connectivity of spaces are related because the topological connectivity necessitates a geometrical relation. This has an important link with how heights are recorded in the framework. Section 3.5.1 explains how *relative* heights can be described where appropriate. It is the *relative* height difference between surfaces (step heights, kerb heights and headroom) which have implications for the topological connection between spaces such that access is provided to a pedestrian.

Pedestrians come in different shapes and sizes and have different needs, expectations and capabilities. Characteristics such as these, affect the way in which they are *able* to move through the space and whether they are *likely* to. The framework will only be concerned with pedestrians' capabilities rather than their behaviour. Some of these considerations are reviewed in section 4.7.2.

Particularly because the description of accessibility must apply inside and outside buildings, the presence of doors imposes many restrictions on the movement of pedestrians. Conceptualisations of doors and other real-world features which affect access are dealt with by the 'feature conceptualisation' design principle (section 3.1). Features which affect access include doors, windows, lifts and staircases. Doors in particular are designed to both facilitate and restrict access. Their geometry affects the rate of movement of people through the door and may restrict the movement of larger people or people carrying bulky equipment. Many doors can be locked and a door may only allow access to people carrying a key or those accompanying somebody carrying a key. Doors may allow access only to those with electronic or magnetic keys or cards. Turnstiles may be used to allow only one person through per key or card and are likely to impose further geometrical restrictions on access. Windows are not usually intended for access (French windows are an exception), but might be used for access in an emergency or for unauthorised access.

In summary, this section has argued that pedestrian access is an important aspect of the built environment and has shown the potential benefits of incorporating it into a description of the built environment.

3.4 Data repository and incremental updating

This design principle states that the implementation underpinning the framework should act as a *data repository* for information, as it becomes available. Thus, by implication, it will hold an incomplete set of data. As data become available, they will be able be added incrementally. As explained below, this applies in terms of the *spatial resolution* and the *density of surveyed height data points.*

3.4.1 Geometry: spatial resolution

Instead of the requirement for a complete resurvey, it is proposed that existing large-scale digital vector topographic mapping data be used as a basis, which only covers 'outside space'. As data become available for mapping particular areas at higher spatial resolutions, i.e. the interiors of particular buildings, these data will be able to be incorporated in an incremental fashion. As was the case with Nolli's map of Rome, it is only appropriate to model the insides of *certain* buildings.

Over time, interior spaces within appropriate buildings (such as indoor shopping centres) will be incorporated, the spaces being seamlessly connected to exterior space. This would enable a data product to be available earlier than if its release was dependent on a complete resurvey.

3.4.2 Geometry: density of height data points

In many 3D conventional data models, a complete and fully-resolved set of 3D co-ordinates is required. This assumes that fully resolved 3D data have been surveyed. This thesis argues that there are pragmatic reasons why an exhaustive 3D data collection exercise should not be required, in order for a populated implementation of the framework to be realised. The dynamic 'as built/as exists' nature of the real world (as opposed to the 'as designed' environment) is difficult and expensive to survey, especially building interiors which cannot be surveyed remotely. It is proposed that the framework be able, initially, to accept a minimal set of height data (e.g. existing spot height information, step and kerb heights and average building heights). In a conventional geometrical model which requires fully-resolved 3D data, heights must be allocated to all 2D coordinates in order to transform them to 3D coordinates. Allocated heights would likely be interpolated or otherwise derived, e.g. from a surface on which the 2D coordinate topologically lies. In data repository, no interpolated value or value which can be derived from existing data should be

stored – it should be interpolated when needed. There is thus a separation between data *as surveyed* and data *as derived* (e.g. interpolated).

From this *description* of data (rather than a fully resolved set of 3D data), a '3D geometry reasoner' (section 3.5.2) will be able to use this minimal information and its own set of rules to generate a 3D geometry. As more height data are collected, they can be added incrementally. As more data are added, the '3D geometry reasoner' will be able to provide better 3D models. The effect of this is a set of data at different spatial resolutions and different certainties about height. Algorithms could be designed to assess height certainty based on the density of height information. More detail about 3D geometry in the framework is provided in section 3.5.

3.4.3 Low data storage requirements

The other characteristic of the data repository is that its representation should be compact, which is one of the reasons why no derivable data should be stored; as with databases in general, the other reason is to do with data consistency. The implementation should be able to derive the data it needs to perform operations such as the incorporation of incrementally added data and the 3D geometry reasoner's geometry, but these derived data should not be stored as part of the repository. Since the implementation does not hold a full 3D model, any 3D operation required must first use the 3D geometry reasoner to generate a geometry. This might be quite a slow process.

3.5 Representation of 2D and 3D geometries

It is clear that many of the applications which this framework strives to support require some degree of 3D geometrical description. However, it is argued in this chapter that the approach of CAD software (section 2.6.1) in which 3D geometry is completely and precisely modelled, is inappropriate for the types of applications being targeted. This is because (as noted at the beginning of this chapter) the framework is concerned with the identification, identity and extent of functional spaces in the built environment and the connections between them; different priorities from those systems which concentrate on pure 3D geometry. These differences are also represented by the difference in approaches between GIS and CAD respectively; this was discussed in section 2.6.3.

As reviewed, many countries already have good two-dimensional topographic base maps available. Some of the objects in topographic base mapping – for example, land parcels – have no meaningful vertical extent. Stoter (2004, ch9) recognises that a mixture of 2D extents (e.g. land parcels and existing registrations) and 3D extents (e.g. flats) need to be accommodated.

Section 3.4 describes the 'data repository and incremental updating' design principle. This requires support for coping with data incompleteness and support for the incremental updating of the dataset as more data become available. It proposes that existing 2D topographic mapping be used as a basis and that this should be supplemented by height data. The reasons for this are:

- large storage requirements of fully resolved 3D models
- pragmatic difficulties in collecting 3D data about the interiors of buildings
- 3D geometry is not the main focus – it is not essential for most of the applications reviewed in chapter 2 (but might be desirable)
- relative heights and the geometry of the connection of spaces are more important for pedestrian accessibility (e.g. steps), including relative heights down to perhaps a one centimetre resolution (e.g. this order of resolution is required for wheelchair users)

It is proposed that fully-resolved 2D data will define the spatial resolution of the model. A set of relative and absolute heights will be incorporated to constrain the geometries along the z axis ('height constraints', section 3.5.1) from which a full 3D geometry can be 'reasoned' (interpolated and extrapolated) by a '3D geometry reasoner' (section 3.5.2) according to a rule set.

3.5.1 Height constraints: absolute and relative heights

As described, the implementation is based on a 2D geometrical model, constrained by height information from which a 3D geometry can be reasoned. This height information can be a mixture of 'absolute heights' (relative to the datum of the coordinate system) and 'relative heights' (relative to the position of a geometry or feature); this is explained further in section 4.5.1.2.

Absolute heights are relative to the main coordinate system. All heights on maps (spot heights and contours) are absolute heights, relative to the map's height datum. Heights in the British National Grid are measured from a surface which passes through the Ordnance Survey Datum at Newlyn, Cornwall[1].

Relative heights are also needed to satisfy the requirements of the framework. Where there is uncertainty in absolute heights, surveyed relative heights become important. Section 3.3.2 identified the problem that relative heights which may be small in magnitude are important factors affecting pedestrian accessibility[2]. It is proposed therefore that all steps should be incorporated as relative heights, because, in the absence of well-resolved height data for the surfaces on either side of a step, it is the *relative height* of the step that is important. The '3D geometry reasoner' (section

[1]Source: http://www.ordnancesurvey.co.uk/oswebsite/freefun/geofacts/geo0274.html

[2]It is likely that airborne scanning equipment would be unable to resolve these heights (LIDAR equipment used by InfoTerra has a stated height accuracy of 10-20cm. *Source: http://www.infoterra.co.uk/lidar_air.htm*).

3.5.2) should be able to compute the surfaces on either side, while honouring the height constraint imposed by the surveyed height of the step. Relative heights can also be used for headroom height under bridges – if the height of either the upper or lower portion of the bridge is refined, the separation between them will be honoured.

3.5.2 '3D geometrical reasoner'

A '3D geometrical reasoner' is needed to produce a 3D geometry from a 2D dataset with an incomplete set of absolute and relative height constraints. As explained in the 'data repository' design principle, only *as surveyed data* are stored. Any 3D *as derived data* are generated by the '3D geometrical reasoner'. From all the data available in the data repository, a 'best guess' about the full 3D geometry is made, making sure that the height constraints are fulfilled. Details of how this is realised in the implementation are given in sections 5.3 and 6.11.

3.6 Representation of time

The world is dynamic, yet many descriptions of the world are snapshots in time. There are some interesting projects looking at the interaction of space and time.

Ordnance Survey's MasterMap (section 2.3.5.2) can be updated by requesting 'change-only updates' (Ordnance Survey, 2005) which only include changes since the last update. The dates used for determining which updates are required can be archived. This gives a rudimentary record of change. Thus, change can be modelled as a by-product of change-only updates.

A framework which models time, not as a by-product of another process, but in its own right would be enormously useful. A record of the time of the real-world appearance, disappearance or change of each feature could be used to reconstruct the model of the state of the real world at any point in time. The complicated issue of the implications of dealing with time in databases and models is reviewed in section 4.8.

In addition, time affects access patterns (section 3.3). Doors and gates of non-residential buildings tend to have permissions for access which are dependent on the time of the day and the time of the week. Patterns of access can therefore change in cycles on a daily and weekly basis (and other time intervals). A formal description is presented in section 5.7.1.2.

3.7 Summary

This chapter has developed a set of specific design principles to guide the development and design of the framework. It is informed by the needs of the targeted

applications reviewed in the previous chapter and some of the technical considerations reviewed in the subsequent chapter.

The framework must be able handle different conceptualisations of features with different identities, properties (attributes) and spatial extents (section 3.1). These must be able to nest and otherwise overlap with other conceptualisations of features. Examples related to buildings have been presented. The framework should be a model of functional spaces in which human activity can take place and such spaces should be treated seamlessly inside and outside of buildings (section 3.2). Information about pedestrian access between these all spaces should be embedded such that the set of spaces accessible by an individual pedestrian can be delineated (section 3.3). These two design principles are illustrated by examples of publicly-accessible parts of the built environment in which there is a complicated set of pedestrian access issues and an unclear distinction between 'inside space' and 'outside space' (which itself is dependent on feature conceptualisations).

The other design principles guide more abstract characteristics of the framework. The 'data repository' design principle (section 3.3) states that the framework should act as a repository to hold all available data as they become available, holding *as surveyed* data rather than *as derived data.* It is argued that there are pragmatic reasons why a full and detailed 3D survey should not be required to allow use of the framework and its implementation. It is also clear from the applications in the previous chapter that some type of 2D and 3D geometrical description is required. The discussion of the representation of 2D and 3D geometry design (section 3.5) specified how incomplete 3D information is dealt with, using absolute and relative height constraints and a '3D geometrical reasoner'. This reasoner produces the *as derived height data* from the *as surveyed data.* Finally, geometry, feature conceptualisations and pedestrian access are all affected by time. Geometry itself, features and their interaction change through time. Pedestrian access is affected not only by this, but by temporal restrictions on the use of spaces and entrances to the spaces.

Chapter 5 develops a conceptual framework for encapsulating these design principles. This is followed by a chapter about a potential implementation of the framework.

Before the chapters on design, a review of the technical issues involved will now be presented.

Part II

Data Modelling

Chapter 4

Data Modelling

This Part is the core of the thesis. From the set of characteristics identified in Part I, a model is developed at various levels of abstraction starting from real-world concepts (high level) towards an implementation in a computer system (low level) and a prototype, demonstrated and reviewed in Part III.

This wide-ranging technical review begins with a review of the art of data modelling. This is followed by more specific reviews of the technical detail of the conceptualisation and representation of geometrical modelling, features, the representation of pedestrian access and the representation of time.

4.1 Data modelling at different levels of abstraction

Data modelling is the process of structuring data about the real or a fictitious world. This process occurs at different levels of abstraction. At a high level – closer to the real world object or system being modelled – it is the process of organising the real world into a well-defined and formalised set of concepts. At a lower level – closer to internal details of a computer system – it is the process of mapping these concepts onto data structures and ultimately to computer memory or disk. The progression from the high level concepts of the real world to the low level digital representation in a computer, necessarily involves abstraction and formalisation. The result of the modelling process is a data model which can be populated with data and which is underpinned by a framework which maps the concepts between various levels of abstraction. As discussed in section 2.1, the more formal and precise the concept definitions are, the greater the potential interpretive power of the resulting system.

Five levels of abstraction are distinguished here, based on Molenaar (1998); Worboys and Duckham (2004). These are listed below (also see figure 4.1).

- The real world – seemingly infinitely complex and impossible to model completely.

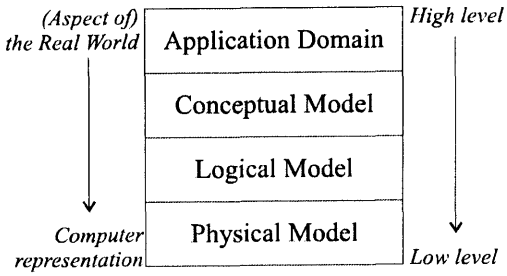


Figure 4.1: The four levels of data modelling (excluding the fifth 'real-world' level).

- Application domain model – an abstraction of the real world; a subset which satisfies the core requirements of one or a set of applications.
- Conceptual model – an implementation-independent formalisation into tangible and well-defined concepts with well-defined relationships between them.
- Logical model – the mapping of concepts and relationships onto system-specific data types used as the basis for an implementation.
- Physical model – the binary representation in memory or on disk handled by the database.

Although these are presented here as well-defined discrete levels, there is no consensus about where the boundaries lie between conceptual, logical and physical models. The convention used here is that widely used in the GIS literature; e.g. Molenaar (1998); Worboys and Duckham (2004). In the database and business world (including at Ordnance Survey), the logical model is system-*independent*, performing the same role as the conceptual model in GIS (though sometimes expressed in terms of general relational database concepts) and the physical model is concerned with all the system-specific detail;e.g. Carkenord (2001).

4.1.1 The real world

The 'real world' is the collection of all facts, whether they are known or not (Kottman, 1999) and whether or not these facts can be conceptualised. It is impossible to know all aspects of the real world and any attempt to try will result in incomplete representation based on a conceptual model.

4.1.2 Application domain

The real world is infinitely complex and it is impossible to have a complete knowledge of all aspects of the real world. However, the task of knowing all there is to know about the real world becomes tractable when considering a subset of that world. The application domain level identifies which aspects of the real world are important for a particular application or set of applications. For example, biologists, ecologists,

chemists, physicists and geologists all study different (but overlapping) aspects of the real-world. Within these disciplines are sub-disciplines.

The creation of any model requires a clear idea of what the model will be used for because this guides the process of abstraction in which irrelevant aspects of the world can be omitted. However, this is usually constrained by difficulty in obtaining data. Often, it is difficult to define the purpose for which the models will be used. This is especially the case for research which is exploratory in nature and also applies to national mapping agencies who provide maps which are fit for a variety of uses. Guidance about how the model should be used is provided by a list of business needs and specifications. The model requirements were defined in Part I.

4.1.3 Conceptual model

The conceptual model abstracts and formalises the subset of the real-world defined by the application domain into well-defined and tangible concepts with attributes and well-defined relationships. These concepts should allow the structure of the system to be communicated in a way that is understandable to non-specialists, while obtaining enough information for their translation into implementation-specific models (Worboys and Duckham, 2004, p56). For example, for a system to track customer orders, the conceptual model would define concepts such as 'customer' and 'product', the relationships between them and the data held by each. Details of how this is to be implemented in a specific system are covered by the logical model.

4.1.4 Logical Models

The logical model describes how the conceptual model is implemented in a particular (computer) system; how high level concepts are mapped onto low level data structures in a particular system.

Logical models can be implemented in database management systems (section 4.2.1), by computer programming languages (section 4.2.2) or by off-the-shelf data analysis software packages such as statistical software and geographical information systems. These all have their own data structures and modelling concepts which affect the design of the logical model. For example, if a relational database is used, then the logical model maps the concepts of the conceptual model to relational database structures and concepts. The database itself handles mapping these to low-level system specific concepts in a way which is (usually) hidden from the user; i.e. the ordinary database users do no need to know anything about the internal file format in which the data are stored, though this may affect performance.

Databases are primarily for data storage and data retrieval and contain tools to support this. Limited summarisation and statistical functionality is also usually available. The other systems listed handle both data storage/retrieval and the modelling of processes. These may use an external database for the storage of data,

particularly where the data volume is high or where concurrent access is needed by multiple users.

4.1.5 Physical models

The physical modelling abstraction level concerns how data and procedures are represented at the lowest levels in a computer system. This is handled by the systems in which the logical models are implemented: DBMSs, off-the-shelf data analysis software packages or custom-built software packages.

4.2 Computer systems

Logical models are implemented in specific computer systems. A computer system comprises hardware, an operating system and a programming environment, database or software product. All these systems have system-specific data structures, concepts and routines and they handle low-level modelling transparently.

As discussed in section 2.1, there is a distinction between models of *state* and models of *process*. Database management systems tend to deal only with state, whereas software applications deal with both state and process. Software applications may also connect to a database for the persistent modelling of state, particularly where the data volume is high or when concurrent access is needed by multiple users.

4.2.1 Database management systems (DBMS)

A database management system (DBMS) is a system whose main (or sole) purpose is to organise data in a systematic fashion, allowing its addition, editing and retrieve via a query language. DBMSs usually have built-in data indexing and tools to enforce the consistency of data[1].

Hierarchical databases and *network databases* are two types of database of which there used to be many examples (Worboys and Duckham, 2004). They have since been superceded by *relational databases*. *Object-oriented databases* are a more recent development, but the immaturity of their development, the computational overhead and lack of standards for object-oriented databases has led to their slow development and poor use. Also, increasingly, relational databases are being extended to allowed support for 'objects'.

4.2.1.1 Relational DBMSs

Relational databases are based on the formal rules originally described by Codd (1970). Data are held in *tables* in rows (*records*) and columns (*fields*). Records can

[1] A summary of databases types is available from Wikipedia (http://en.wikipedia.org/wiki/Databases).

be joined together between tables with *relational joins*. *SQL* is a standard language for relational databases through which data can be entered, edited and retrieved (though there are usually small variations in SQL between relational databases from different vendors).

The data types for attributes in database usually include Booleans, integers, floating point numbers, text strings, dates and BLOBs (binary objects). Spatial data types have now been implemented many relational databases (e.g. Oracle, Postgres and MySQL), with corresponding spatial queries. Many also support the storage of more complicated objects, using an object-oriented-like methodology. Relational databases are by far the most common type of database in use.

4.2.1.2 Object-oriented DBMSs

Object-oriented DBMSs attempt to provide a means for making the logical model closer to the conceptual model. Object-oriented databases emerged in the early 1990s, but they remain an immature technology and a niche product due to their lack of standards, their poor performance (when compared to relational databases) and their complexity.

Objects corresponding to the high-level concepts of a conceptual model can be defined by packaging up a set of data types which include other objects, together with a set of programming routines into a *class*. Any number of *instances of classes* can be created, each of which is an *object*. Unique amongst other types of database is that programming routines as well as the data themselves are part of the database (they are part of a class' definition); thus, object-oriented databases can model *process* as well as *state* without relying on an external application outside the database. In addition, because of this and because classes are so customisable, high-level concepts can be defined directly. It is possible to represent the same high-level concepts in a relational database, but it has to be done by their mapping to more rigid database records.

Some object-oriented databases are in fact *object-relational* in that they present an object-oriented view of the data to the user, but in fact convert the data seamlessly to and from relational database tables.

4.2.2 Programming languages

Programming languages are used to express computer-readable instructions. Different languages exist, for different purposes with different amounts of expressive power. Scripts are generally relatively simple lists of instructions to sequence the execution of different software modules. Full programming languages, give more control to the programmer of the computer platform (access to memory, disk space, the operating system and external devices) and are used to write software modules and software applications. VBA (Visual BASIC for Applications) is a scripting lan-

guage (a simplified version of the programming language BASIC) which is designed to run inside compatible Windows applications, allowing a sequence of operations to be defined and simple user interfaces to be designed. Full programming languages include BASIC, Pascal, C, C++ and Java; these are multipurpose languages, but have differences in design, execution speed and capabilities.

Programming languages provide a range of data types for representing a model state, a range of operators and built-in libraries of functions. There is the facility to use external or user-defined libraries and the facility to connect to external database management systems for the persistent storage of state. The application itself or a library of functions being used by the application, must handle the logical model for mapping between the data structures in the two systems.

All computer languages need to be interpreted at some level into the machine code specific to the hardware of the computer. Some languages are completely compiled with a *compiler* prior to their running (the compiler generates the physical model). Other languages (including all scripting languages) are interpreted as they are run by a *run-time environment* (the run-time environment generates the physical model).

Two decades ago, the majority of programming languages were *procedural languages.* Procedural languages are organised around *sets of procedures which operate on pieces of data.* In the mid-1980s the object-oriented paradigm became the predominant programming methodology (the same methodology as used in object-oriented databases). The object-oriented paradigm is higher level than procedural languages, packaging up state and process into specific *classes* with specific remits. Instances of these classes may be able to operate autonomously or semi-autonomously and may be able to interact with other classes. The processes defined within classes are still procedural with many of the same types of programming structures and data types, the data types having often been implemented as classes themselves. Many languages have now been extended to support the use of objects, which can be used in parallel with the original procedural approach. This even applies to some scripting languages such as PHP and Perl. C++ is a version of C with object-oriented extensions. Java is based on C++, but was designed as an object-oriented language rather than evolving from a procedural language. It is purer in its implementation of object-oriented principles, but nevertheless retains non-object oriented data structures such as numeric primitive data types and arrays, for performance reasons.

4.2.3 Off-the-shelf software

Off-the-shelf software is becoming increasingly customisable. In a Microsoft Windows environment, the COM and .NET frameworks allow interoperability between modules of different software systems and many major software applications exploit such frameworks. Standard scripting languages such as VBA and Python are becoming more powerful with the move away from proprietary macro languages.

With the increasing modularisation of off-the-shelf software using standard (usually operating-system dependent) frameworks, there is an increasing trend to customise existing software rather than writing custom software from scratch.

4.3 Object-oriented programming

The principles of object-oriented design principles are well known (e.g. Bell, 2005). *Data* which represent a system's state, and *procedures* which represent processes occurring within the system, are grouped into *components*, each of which has a well-defined role. These components are called *objects*.

Objects are *instances* of templates called *classes*. Any number of objects can be created from a class (limited only by memory). The class defines *fields* in which the data are stored and *methods* which are sets of procedural instructions which define the behaviour of the class. Fields and methods are usually attached to instances of classes (objects) in which the value of the field and the data upon which the methods operate are specific to the instance of the class; this is independent of other instances of the same class. Some fields and methods, however are *static*, being attached to the class itself rather than to a specific instance. Other than some primitive data types and arrays, all data types and structures are themselves classes. Fields and variables of a class can be assigned different scopes. *Public* fields and methods are accessible from outside the objects by other objects; *private* fields and methods are only accessible from within the class. Thus objects can contain hidden methods used internally and expose themselves to other objects in a well-defined way. Depending on the dependencies between classes, objects can be autonomous or semi-autonomous entities.

Three widely-held principles of object-oriented implementations are (Bell, 2005, p139):

- Encapsulation
- Inherence
- Polymorphism

4.3.1 Encapsulation

Encapsulation hides the internal implementation of a class. It does this by marking methods and fields as either 'public' or 'private'. The set of 'public' methods effectively provides an interface for interaction from outside the class. This enables the internal state of an object to be more easily managed and makes it possible to modify the internal representation (private methods and fields) without changing the way in which other objects interact with it. Other objects do not need to know about the internal details relevant only to the functioning of the class. In this way, a class can encompass two levels of data abstraction.

4.3.2 Inheritance

Classes are all placed within an object hierarchy. Classes *inherit* the fields and methods from their *superclasses* (ancestors). *Subclasses* (descendents) are *specialisations* of their superclasses, because they inherit fields and methods to which more can be added and they may additionally *override* some inherited methods, replacing or adding to the class' behaviour. An *abstract class* is a class which is present purely for inheritance reasons and cannot have instances of it created. Other classes are called *concrete classes.*

For example, consider a 'coordinate' class. An abstract coordinate class with fields and methods common to all coordinates (e.g. an on-screen display colour) can be defined. No instance of this can be created, but it can be the basis of a class hierarchy. If a concrete 2D coordinate class is made as a subclass of the abstract class, 'x' and 'y' fields can be added, in order to describe its position in 2D space. The 2D coordinate class can be considered to have an 'is a' relationship with the abstract coordinate, i.e. it is a 'coordinate'. A 3D variant could be a subclass of the 2D variant, inheriting the 'x' and 'y' fields, and further adding a 'z' field. The methods are inherited, and their behaviour can be replaced to be added to by overriding the particular methods.

The choice of hierarchy depends on the implementation.

4.3.3 Interfaces

Interfaces define 'public' methods (the methods' names, parameters and return values), *but do not implement them.* Classes may optionally implement one or more interfaces; each interface inherited must have all its methods implemented. Interfaces themselves can also be organised into a hierarchy similar to a class hierarchy.

4.3.4 Polymorphism

Polymorphism arises when an inherited method of a class is *overridden* to modify or add to its behaviour. When the method of an object is called, its behaviour depends on the instance's class. The effect of polymorphism is that the same apparent method on each object of a set, may exhibit a different behaviour from each other (depending on the class of which they are instances).

For example, consider a 'getPerimeter' method, used to find the perimeter of any of instance of a 'polygon' class or a 'circle' class. The calculation used to calculate the perimeter of polygons and circles differs, so the implementation of the 'getPerimeter' method for the 'polygon' class and the 'circle' class will be different. When the 'getPerimeter' method of the instance is called, the implementation used will depend on the class of which the instance is. The user does not need to know this, only that the 'getPerimeter' method will return the correct answer.

4.3.5 The wider applicability of object-orientation

The process of abstraction encourages the formalisation of a system into components. The highest-level abstractions conceptualise a system as a set of tangible real-world components (the conceptual model), whereas the lowest-level abstractions (more to do with implementation) are concerned with components of the computer system and storage media in which the model is implemented.

The classes in object-oriented design provide a convenient way of packaging up high-level concepts into components. These components have well-defined and specific remits and they can interact with each other through the use of public methods. Inheritance, encapsulation and polymorphism provide powerful tools to organise the components. In particular, the principle of encapsulation and visibility allows unimportant internal details to be hidden, exposing only methods which are directly relevant to the remit of the class.

As discussed, these principles are in widespread use in object-oriented programming (they have been applied to databases, but with less success). Since object-oriented principles are powerful tools with which to abstract and implement systems, it is convenient to also use the same principles to express the conceptual model.

UML (next section) is a modelling language, based on the rich vocabulary of object orientation. In this thesis, it will be used to express both the conceptual and logical models.

4.4 Unified modelling language (UML)

The Universal Modelling Language (UML)[2] has emerged as a widely-used standard notation for designing and describing systems using object-oriented principles (Miller, 2004). Although UML class diagrams commonly used to express classes in object-oriented logical models, they have also been used to replace entity-relationship diagrams for conceptual models (Urban and Dietrich, 2002; Maksimchuk and Naiburg, 2003).

There are several different types of UML diagram which describe different aspects of a system. A good web resource for UML can be found at http://bdn.borland.com/article/0,1410,31863,00.html. The majority of the diagrams used in this thesis are *UML class diagrams* and *UML sequence diagrams.* The use of free text in all UML diagrams is encouraged to provide further detail.

4.4.1 UML class diagrams

Class diagrams are used to show the design of classes and how they relate. An annotated example is illustrated in figure 4.2.

[2]http://www.omg.org/gettingstarted/what_is_uml.htm

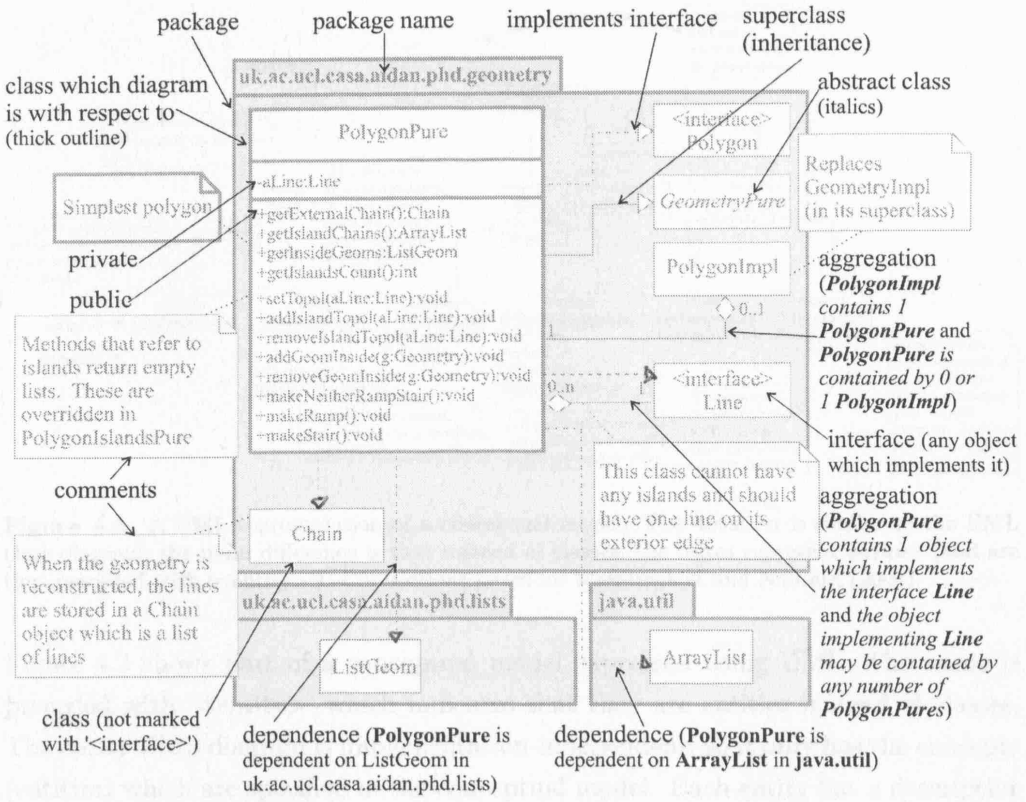


Figure 4.2: An annotated example of a UML class diagram. The class being described in 'PolygonPure' (with a heavier outline) in the 'uk.ac.ucl.casa.aidan.phd.geometry' package, which is a subclass of the abstract class 'GeometryPure' (solid line and open arrow head) and implements the 'Polygon' interface (open arrow head). The other classes shown as direct dependents, whose relationships with the class of focus are indicated by the symbology of the connecting lines. Their details are not shown.

A class is represented by a rectangle. This can be divided into three parts if the fields and methods need to be shown; if this is the case, the first contains the name of the class, the second lists the fields and the third lists the methods. The data types of fields, method parameters and return values are indicated after the colon. '+' and '-' symbols indicate their scope and whether they are public or private, respectively. Many of the diagrams presented here are focused on one class (outlined in a heavier line) and whose dependencies are shown. Interfaces have their names preceded with the word '<interface>', and the boxes representing the interfaces divided into two parts rather than three (interfaces do not have fields).

Relationships with other classes and interfaces are shown as different types of arrows, examples of which are shown in the figure. The implementation of an interface is shown by a dotted line with an open arrow. Inheritance is shown as a solid line with an open arrow. Aggregation is shown by a solid line with an open diamond shape touching the class which is aggregated and is labelled with the valency (aggregation quantity) or valency range. Direct dependences are shown by dashed arrows.

Comments are indicated in rectangles with the corner folded over.

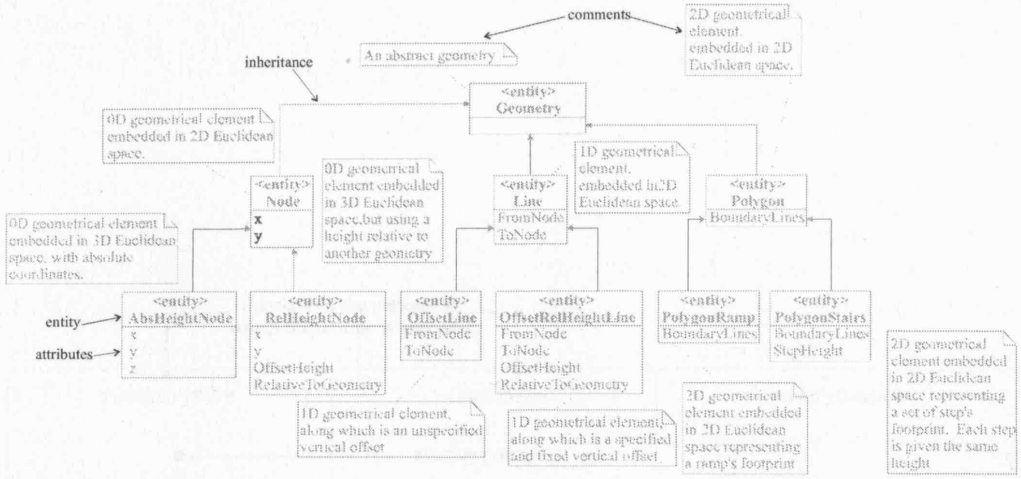


Figure 4.3: A UML representation of a conceptual model. The notation is similar to the UML class diagram; the main difference is that instead of classes, the boxes represent *entities* (and are thus preceded with '<entity>'). Conventions based on Maksimchuk and Naiburg (2003).

Figure 4.3 shows part of a conceptual model expressed using UML. The name is preceded with '<entity>' which indicates that they are entities instead of classes. The entity UML diagram is implementation-independent, so it only has the concepts (entities) which are specified in the conceptual model. Each entity has a descriptive set of attributes. Inheritance is only shown where it makes sense at a conceptual model level; for example, in figure 4.3, there is a hierarchy of geometry types. This makes sense at a conceptual level, i.e. a 'HeightNode' is a specialisation of a 'Node', but an implementation may have a different hierarchy.

4.4.2 UML sequence diagrams

Figure 4.4 shows an annotated example of a UML sequence diagram. This is essentially a flowchart of how a procedure or algorithm works across different classes. See the figure for an explanation.

4.5 Geometry

This section will review conceptual and logical models for the representation and handling of geometry and space. 'Space' can be considered in a variety of ways. In this thesis, the quantitative surveyor's view of space will be used; that space is a physical extent which can be quantitatively described according to a Euclidean metric.

This section will first introduce the concept of Euclidean metric space which is a framework for conceptualisating space. It will then distinguish between two approaches to modelling geometry within space (space-filling and non-space-filling), introduce the concept of topology (spatial relations) and then review different geo-

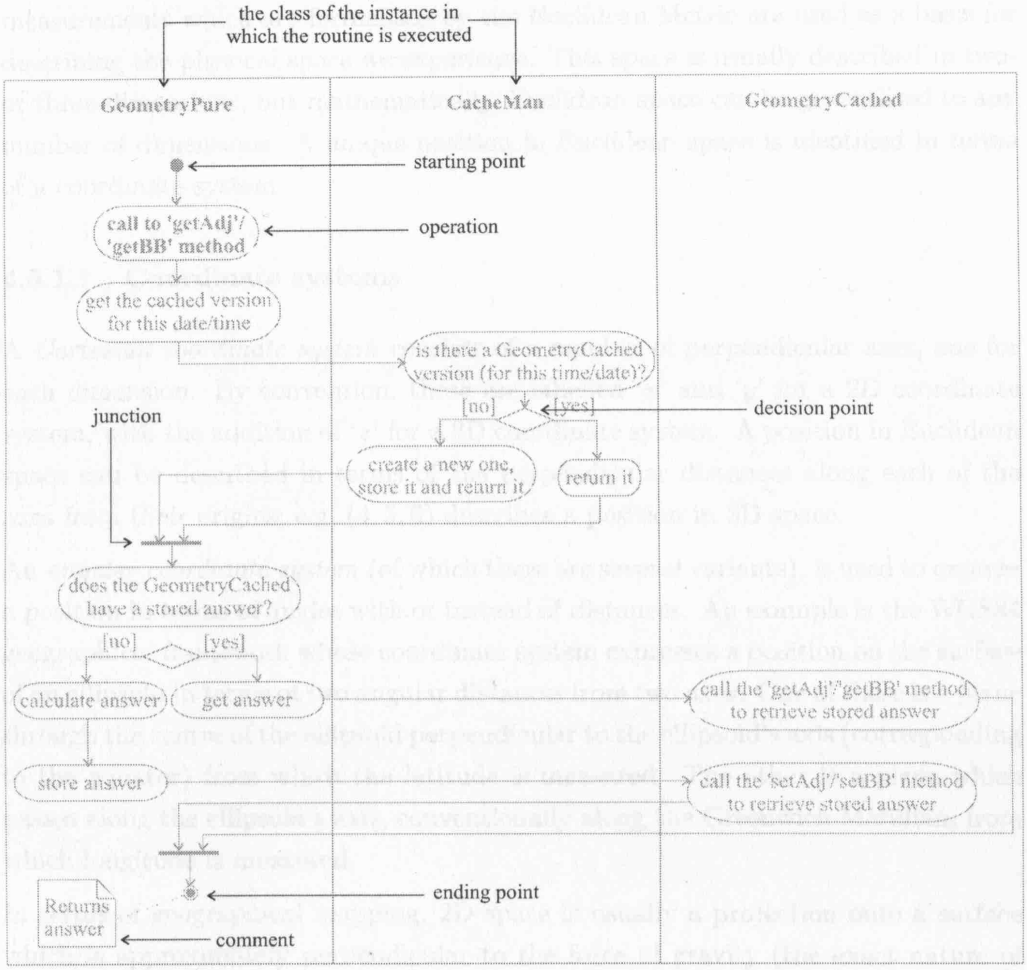


Figure 4.4: An annotated example of a UML sequence diagram. The procedure starts at the top left black dot and the participating classes are arranged horizontally along the top of the diagram. Solid lines link operations indicated inside rounded rectangular shapes, all of which are in the column corresponding to the class in which the specific operation is executed. Decisions are indicated by rhombus shapes and junctions in the flow of the procedure are indicated by solid small narrow horizontal rectangles. The procedure ends with a dot inside a circle. As with the class diagram, comments are displayed in rectangles with folded-over corners.

metrical models. The final part of this section will reflect on these different approaches and apply them to the needs of the framework.

4.5.1 Metric space

The quantitative description of space requires a framework incorporating the concept of *distance*. A *metric space* is such a framework (Worboys and Duckham, 2004). The measurements which are formalised by the Euclidean Metric are used as a basis for describing the physical space we experience. This space is usually described in two- or three-dimensions, but mathematically, Euclidean space can be generalised to any number of dimensions. A unique position in Euclidean space is identified in terms of a coordinate system.

4.5.1.1 Coordinate systems

A *Cartesian coordinate system* consists of a number of perpendicular axes, one for each dimension. By convention, these are labelled 'x' and 'y' for a 2D coordinate system, with the addition of 'z' for a 3D coordinate system. A position in Euclidean space can be described in terms of the perpendicular distances along each of the axes from their origins; e.g. $(4, 5, 6)$ describes a position in 3D space.

An *angular coordinate system* (of which there are several variants), is used to express a position in terms of angles with or instead of distances. An example is the WGS84 geographical framework whose coordinate system expresses a position on the surface of an ellipsoid in terms of two angular distances from two axes. One of these is a plane through the centre of the ellipsoid perpendicular to the ellipsoid's axis (corresponding to the equator) from which the latitude is measured. The other is a plane which passes along the ellipsoid's axis, conventionally along the Greenwich Meridian, from which longitude is measured.

In terms of geographical mapping, 2D space is usually a projection onto a surface which is approximately perpendicular to the force of gravity (the exact nature of this surface, coordinates and method of projection is defined by the geographical framework). Most regional or national topographic mapping uses a locally optimised 2D Cartesian coordinate system. These usually incorporate the concept of height from a datum (e.g. Ordnance Survey Datum at Newlyn, Cornwall[3]), which are used for contours and spot heights. Thus a 3D position can usually be described in a geometrical framework.

4.5.1.2 Absolute and relative coordinate systems

A geographical framework is usually based on one metric space with one coordinate system. However, as explained in section 2.3, different coordinate systems tend to

[3]Source: http://www.ordnancesurvey.co.uk/oswebsite/freefun/geofacts/geo0274.html

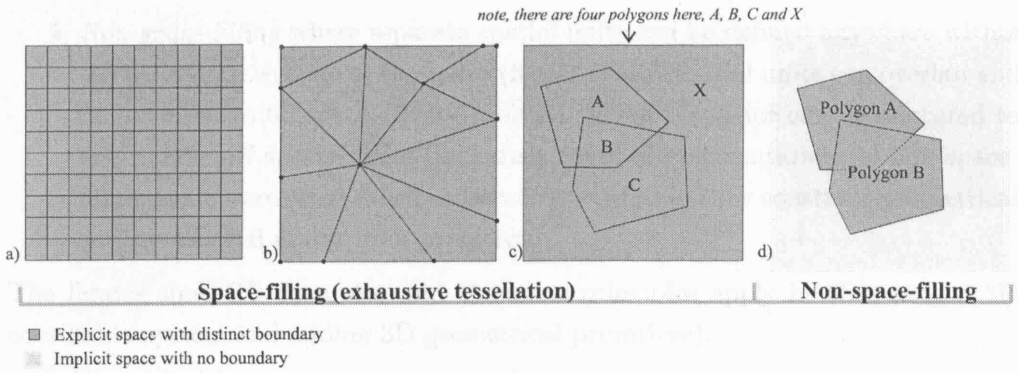


Figure 4.5: Approaches to geometrical modelling, shown for 2D space (but which apply equally to 3D space). *Space-filling* representations tessellate space regularly (a) or irregularly (b and c) into spatial units (tiles in 2D or volumes in 3D), in which every position within the space can be allocated to exactly one spatial unit. *Non-space-filling* representations (d) do not exhaustively tessellate the space; instead spatial units can be defined anywhere in the space and are allowed to overlap. The extent of the space is dependent only on the size of the coordinate system. Every position within the space can be allocated to any number of spatial units (including 0).

be used for mapping in different parts of the world and for different coverages of data because of better local fit. As long as two coordinate systems are known and the transformation from one to the other is known, a GIS will seamlessly convert between them (though transforming between coordinate systems may result in a loss of precision).

As shown by various CAD tutorials[4], many CAD and geometrical modelling systems support the description of entities with reference to different local coordinate systems which are embedded in the main coordinate system. For example a complicated and intricate geometrical object may be defined on a local coordinate system, which is then placed as a group in the main coordinate space. Thus the absolute position of the individual geometry of the individual feature is relative to the main coordinate system.

4.5.2 Representations of space

Space is continuous with no natural units. In order to describe space quantitatively, either it needs to have a function fitted to it (to describe a continuous phenomenon) or it needs to be broken down into discrete spatial units. There are two approaches to dealing with discrete space: space-filling and non-space-filling:

- *Space-filling* where the 2D space is exhaustively tessellated with a regular (figure 4.5a) or irregular tessellation (figure 4.5b and figure 4.5c). Every position within the space can be allocated to at least one spatial unit. Figure 4.5c shows an irregular tessellation into four spatial units. Implementations of space-filling models are often called *decomposition models* as they decompose space into spatial units.

[4]For example, one offered by AutoDesk for its AutoCAD software, obtainable from http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=3028436&linkID=2475176

- *Non-space-filling* where separate spatial units can be defined anywhere within the coordinate system of the space (figure 4.5d). Spatial units can overlap and do not need to fill space. Every position within the space can be allocated to any number of spatial units (including none). Implementations of non-space-filling models are often called *constructive models* as they construct geometrical entities (spatial units) from primitives.

The figures show 2D examples, but the same principles apply in 3D (using a 3D coordinate system and adding 3D geometrical primitives).

4.5.2.1 Space-filling representations

The property that every position within the space can be allocated to a single spatial unit, makes space-filling representations suitable for modelling some aspect of space itself, rather than features within space. Space-filling representations are often used for modelling *field data types*, in which a value varies continuously over the whole spatial extent. Two popular ways of doing this are through the use of *rasters* which apply in both 2D and 3D space, and triangular irregular networks which apply in 2D space. Tetrahedral networks can be used in 3D space (Pilouk, 1996).

Rasters tessellate space into *regular* spatial units (figure 4.5a). Each spatial unit is allocated a field value which is of some representative value (e.g. height) for the extent of the spatial unit. Triangular irregular networks (TINs) tessellate space *irregularly* into 2D triangles (figure 4.7b). TINs have topological properties which can be used in applications requiring topological analysis[5]. TINs can also represent surfaces if the vertices of the triangles are given values which are displayed as heights. Although TINs whose vertices are full 3D coordinates which are fully closed around volumes of 3D space, are widely used in 3D model for representing objects, the TINs referred to in this thesis are based on 2D TINs. 2D TINs are space-filling in 2D space. Where vertices are given heights, they represent surfaces with the restriction that no part self-overlaps, as seen from above. This type of surface is described as a '2.5D surface' because the unequal treatment of the three dimensions and the representational flexibility this imposes precludes their classification as 3D.

The representational disadvantage of regular tessellations over irregular tessellations, is that regular discretisations are not able to preserve positional precision – instead positions are snapped to the grid upon which the regular tessellation is based. In digital elevation modelling, irregular spatial units can be defined such that the boundaries coincide with relevant discrete entities, such as ridges, valley bottoms or other breaks of slope). As shown in figure 4.6, this has important implications for landscape modelling (Jones *et al.*, 1990; Braun and Sambridge, 1997; Tucker *et al.*, 2001; Slingsby, 2003; Rousseaux, 2003). TIN edges which have been fixed to coincide with breaks of slope are known as *breaklines* and TINs for which this has been done are said to be *constrained.* In addition, the representational resolution

[5]http://www.voronoi.com/

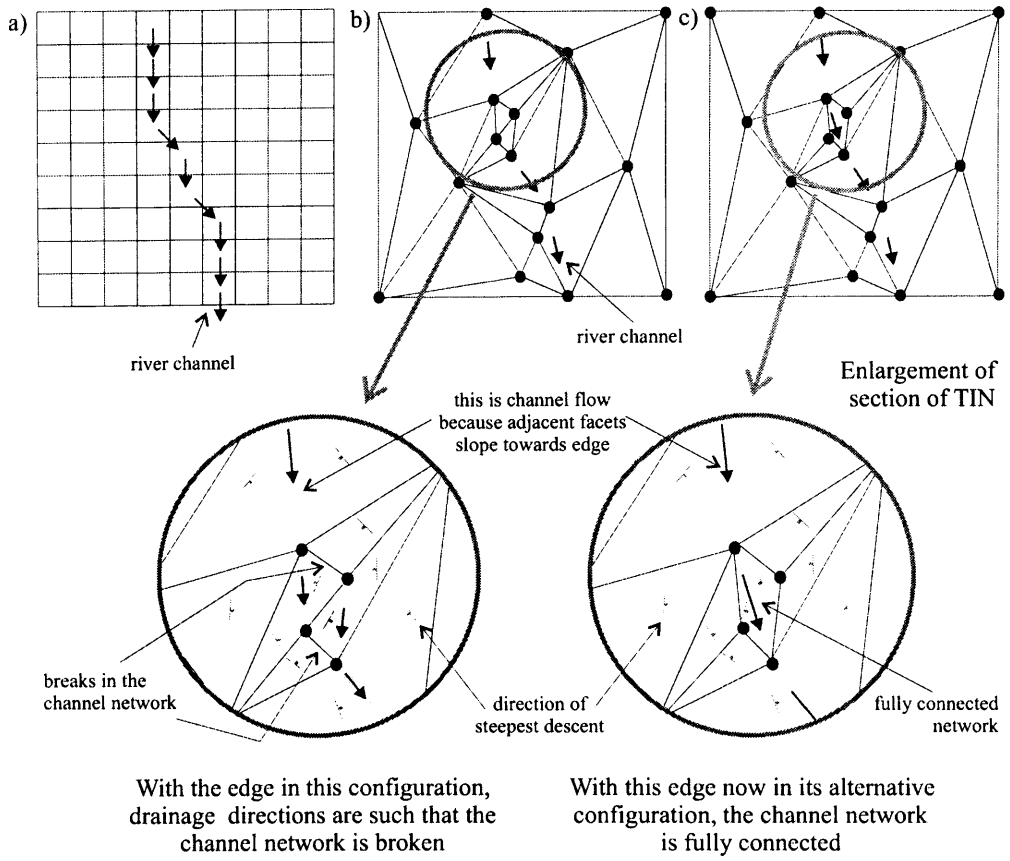


Figure 4.6: An illustration of the fact that alternative triangulations yield different surface morphologies. *Source: Slingsby (2003, figure 3)*

of irregular tessellations can be changed to take into account the amount of spatial autocorrelation. In comparison, rasters destroy any spatial structure below the fixed resolution of the raster. However, they are in widespread use, because of digital acquisition methods, their simplicity of structure and the computational efficiency of raster-based algorithms. They are also suitable where the spatial structure of data is regular, unknown or unimportant. (Figure 4.5c shows an irregular discretisation which is not based on triangles.)

4.5.2.2 Non-space-filling representations

Non-space-filling representations do not have the requirement to fill space (as indicated by the name). Geometry can be placed anywhere within the coordinate system of the space, regardless of the position of other geometries. These are appropriate representations for discrete real-world entities which may overlap and which do not necessarily completely cover the space in which they exist. Non-space-filling representations have greater representational flexibility. Since geometrical entities (spatial units) do not necessarily tessellate, their geometries do not need to be dependent on any of the other geometrical entities. ESRI's Shapefile format is based on this type of model.

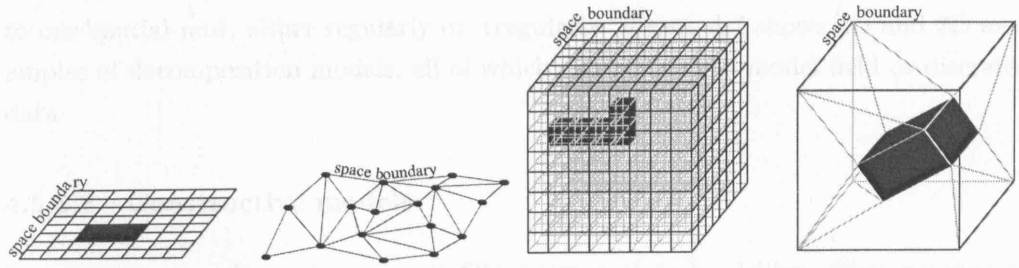


Figure 4.7: Decomposition models: examples of 2D and 3D space-filling geometrical models. The examples shown are modelling discrete data (indicated by the black shaded areas) with the exception of the triangular irregular network which is modelling a field data type.

4.5.2.3 Discussion

Different approaches to geometrical modelling have different advantages and disadvantages. The most suitable approach is dependent on the nature of the phenomenon being modelled, the nature of the data available to populate the model, and the intended use of the data.

For the polygon features in the Topographic Layer of OS MasterMap (section 2.3.5.2), a 2D space-filling model which uses an irregular tessellation (figure 4.5) is most suitable, because the polygons are not regular, non-overlapping, and they exhaustively cover the space. However, this connection between non-overlapping spatial units (polygons) and real-world features means that overlapping or nested features cannot be described. The 'feature conceptualisations' design principle (section 3.1) explained the need to be able to describe an almost infinite range of features with different extents which overlap in non-simple ways. In terms of representation, it would appear than the non-space-filling approach is a more suitable representation. However, this section discusses the pure geometrical aspects of the model and feature conceptualisations are not considered here.

4.5.3 Representations of geometry

The space-filling and non-space-filling approaches to modelling space are ways in which space is discretised. Discrete space has a geometry which needs to be described. In general, space-filling approaches are used to attach attributes to properties of space itself, whereas non-space-filling approaches define the geometry of objects in the space. The types of geometrical models which correspond to these are known as decomposition models and constructive models, respectively.

4.5.3.1 Decomposition models

Decomposition models are a space-filling approach to modelling space. They exhaustively decompose space into spatial units such that any position in space belongs

to one spatial unit, either regularly or irregularly. Figure 4.7 shows 2D and 3D examples of decomposition models, all of which can be used to model field or discrete data.

4.5.3.2 Constructive models

Constructive models are a non-space-filling approach to modelling. They construct geometrical entities (spatial units) from primitives. As shown in figure 4.8, there is a wide variety of methods for constructing geometrical entities, with different complexities of primitives.

Mathematically, half-space models (figure 4.8a) are the purest form of constructive model (Mäntlyä, 1988). A hyperplane[6] (an infinite line in 2D; an infinite surface in 3D), which can be described by a function, divides space into two half-spaces. The intersection of multiple of half-spaces, such that a fully bounded space results, can be used to describe a solid. Thompson (2006) shows that the functional description used has advantages for numerical stability and robustness. Figure 4.8a illustrates the description of a 3D cylinder using three hyperplanes (an infinite, open-ended cylinder and two infinite planes). Primitives in half-space models are half-spaces (defined by hyperplanes).

Vector geometrical models (figure 4.8b) are generally line-based models, in which polygons are represented by their boundaries. The 3D equivalents are known as *boundary representation* models ('b-rep'), because volumes are represented by their boundaries, setting them apart from other constructive representations. Primitives are all or some of 0D points, 1D line segments (which may include curved line segments), 2D (triangles or other polygons) and 3D volumes (tetrahedra or other polyhedra). Each 3D object in a boundary-representation model may be modelled by a space-filling topologically-structured 2-manifold model (section 4.5.4.1).

Primitive instancing or *parametric* methods (figure 4.8c) use higher-level geometrical primitives defined in a primitive library. Their positions, orientations and dimensions can usually be controlled using primitives. The flexibility of geometrical representation depends on the primitives defined in the library. Geometrical primitives in the library can be of any dimension, embedded in the 2D or 3D Euclidean framework being used.

Constructive geometry (figure 4.8d) is similar to primitive instancing, but the flexibility of representation is improved by the ability to define geometries by the combination of primitives using set operators such as *intersection* ($\cap$), *difference* ($\setminus$) and *union* ($\cup$). These models apply both in 2D and 3D; in 3D, they are more commonly known as *constructive solid geometry (CSG)*.

Sweep methods (figure 4.8e) are those which define a geometry as the area or volume through which a geometry is swept along a path. Sweeping a line (1D) along a path

[6] http://en.wikipedia.org/wiki/Hyperplane

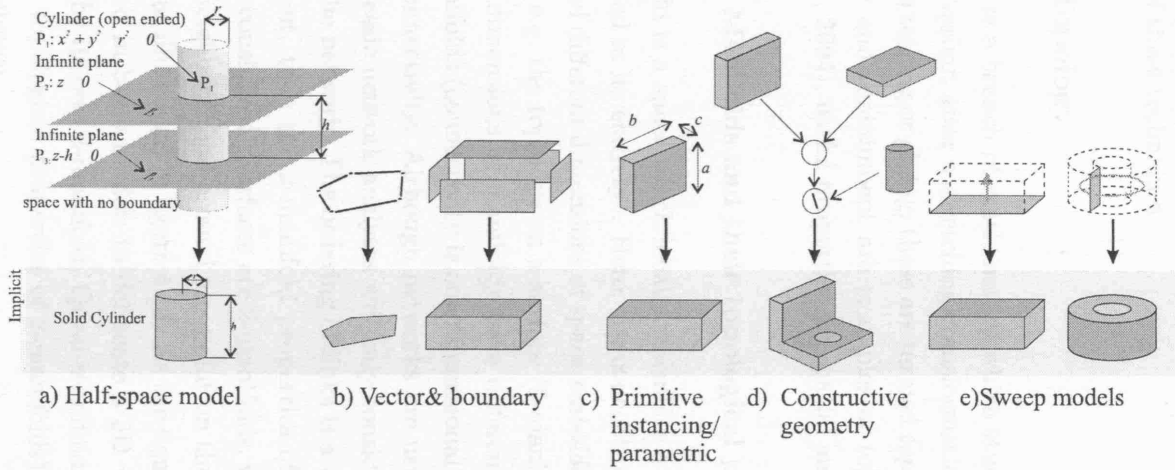


Figure 4.8: Constructive models: examples of 2D and 3D non-space-filling geometrical models which construct geometrical entities from primitives which are often controlled by parameters. *(a)* A solid geometry in a half-space model is the closed space resulting from the intersection of two or more half spaces (unbound spaces). Half-spaces are described by hyperplanes, where a hyperplane divides space into two half-spaces (unbound spaces). Linear geometries can be defined by the intersection of hyperplanes. *(b)* Vector models describe geometry using all or some of the basic 0D, 1D, 2D and 3D primitives of points (coordinates), lines (straight or curved line segments), polygons (planar or curved) and volumes. In 2D models, polygons are usually described by the lines that define the boundary. In 3D models, volumes are usually described by the polygons which bound it. The term 'boundary-representation' or 'b-rep' is usually used in a 3D context. *(c)* Primitive instancing is where a library of solid geometries is already defined, whose position, orientation and dimensions can usually be controlled through the use of primitives. *(d)* Constructive geometry is like primitive instancing, except that instances can be combined through set operators such as 'union' (∪) and 'difference' (\) to create new geometries. *(e)* Sweep models define a new geometry as the space through which an existing geometry is swept. In this example, a cuboid is defined by sweeping a polygon in the direction shown by the grey arrow.

will produce an area (2D) and sweeping an area along a path will produce a volume or solid. The geometry of objects carved with a lathe can be modelled using this approach; the 2D geometry along the radius perpendicular to the axis of the lathe can be swept around that axis. A specific version of a sweep method is often used in '2.5D discrete' models (section 4.5.5.1), in which a horizontal 2D geometry is swept along a perpendicular vertical axis.

Geometrical modelling software usually allows the user to define geometries using a number of these techniques.

4.5.4 Topology

Topology is a branch of mathematics which studies the spatial relationships which remain invariant after a stretching transformation is applied to geometry, provided there is no tearing or gluing; these are termed *topological relationships.* Connectivity, adjacency and containment are examples of topological properties (Worboys and Duckham, 2004), useful properties for spatial analysis.

4.5.4.1 Manifolds and their topological properties

A manifold is a space which locally resembles a Euclidean space, but may be more complicated in its entirety[7]. Here, the concept of a manifold is used to consider the topology of different dimensions of space *embedded within* a higher dimensional overall space; e.g. the topology on a surface (2-manifold) which exists within 3D space. Different dimensions of manifolds have different topological properties. Networks are 1-manifolds (connectivity is one-dimensional) whose only topological property is that of connectivity. Although networks are usually depicted in 2D space (e.g. on paper), classic network analyses will only consider the 1-manifold topological properties of the network. The ordering of links is a topological property of 2-manifolds; if important, then the 2-manifold properties of the network's depiction on paper should be considered. Surfaces are 2-manifolds, which have the topological property that the neighbourhood around any point on the surface is topologically equivalent to a disc, as illustrated in figure 4.13. As the figure also shows, 2-manifolds are commonly used in b-rep models to delineate a 3D volume of space, each object being modelled by its own 2-manifold. Quasi-manifolds and non-manifold surfaces (which break the topological properties of 2-manifolds) may be used for more representational flexibility.

In decomposition models (space-filling models which partition all of space into a set of tessellating and non-overlapping spaces), all the space can be considered to be in the same manifold. Figure 4.9 shows that the 2-manifold topology (adjacency) of regular tessellations is implicit and easy to resolve. The topology of irregular

[7] http://en.wikipedia.org/wiki/ManiFold

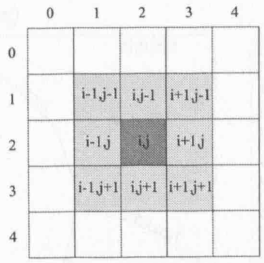


Figure 4.9: The topology of regular tessellations. The built-in primitive topology allows the adjacent cells to be derived simply from the cell position.

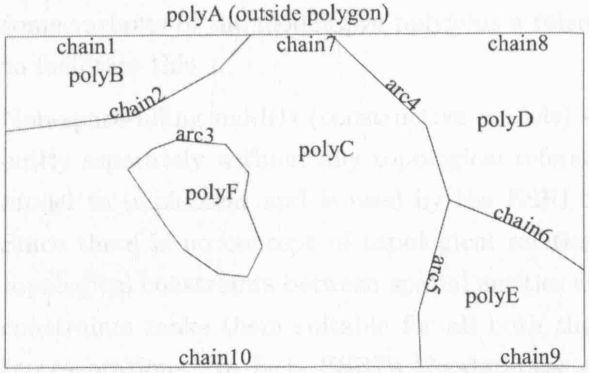
tessellations is not implicit; these must be computed when required, if they are not already held by the model.

In constructive models (non-space-filling models where spatial entities are constructed independently of each other), each spatial entity can be considered to be in its own manifold, all of which are embedded in a common space. For example, Polygon A and Polygon B in figure 4.5d are both separate 2-manifolds which are embedded in 2D space. The topological properties *between* these manifolds are much more complicated than *within* a single manifold. 'Overlap' exists as an additional topological relationship, and all topological relationships can have *partial* relationships. The 4- and 9-intersection models are frameworks for describing 2D topological relationships between pairs of spatial entities, both of which are reviewed and compared by Egenhofer *et al.* (1993). The 4-intersection model considers the interior and boundary of each spatial entity. A two-by-two matrix is used to describe whether there is an overlapping relation between the four combinations of the interior and boundary of the first entity and the interior and boundary of the second entity. This results in $2^4 = 16$ possible spatial relations which have been given names including 'disjoint', 'contains', 'inside', 'equal', 'meet', 'covers', 'covered by' and 'overlap' (Egenhofer *et al.*, 1993). The 9-intersection model additionally includes the spatial entities' exteriors, resulting in a three-by-three matrix with the ability to recognise $2^9 = 512$ spatial relationships (Egenhofer and Herring, 1994). However not all the spatial relationships are possible in reality and many are topologically equivalent (Billen and Zlatanova, 2001). Nevertheless, the 9-intersection is well-used; for example, the APIs provided with ESRI ArcGIS allow spatial queries to be expressed in terms of the 9-intersection model.

These topological relationships can be stored for all pairs of objects in a matrix. In 3D, there is an even greater range of topological relationships (Zlatanova, 2000; Billen and Zlatanova, 2001; Ellul and Haklay, 2005).

4.5.4.2 Role of topology in vector geometrical models

Vector geometrical models which model geometry within the same manifold can use topological properties for validation. 2-manifolds have a well-defined relationship between 0D, 1D and 2D primitives which is expressed by the Euler-Poincaré formula



Chain	Coordinates	L-R Poly
1	(x,y),(x,y),(x,y)...	A B
2	(x,y),(x,y),(x,y)...	B C
3	(x,y),(x,y),(x,y)...	C F
4	(x,y),(x,y),(x,y)...	D C
5	(x,y),(x,y),(x,y)...	E C
6	(x,y),(x,y),(x,y)...	D E
7	(x,y),(x,y),(x,y)...	A C
8	(x,y),(x,y),(x,y)...	A D
9	(x,y),(x,y),(x,y)...	A E
10	(x,y),(x,y),(x,y)...	A C

Figure 4.10: An illustration of a 2D topologically-structured model with planar enforcement. Chains (ordered sequences of line segments) are described by sequences of coordinates and have the property that each has exactly two adjacent polygons along its length. Polygons can be built from these chains. This type of structure is well known; see for example, Worboys and Duckham (2004, p180).

(Hoffmann, 1989) (which has been generalised to take account of holes) of $V - E + F - 2(1 - G) = 0$ (where V is the number of nodes, E is the number of lines, F is the number of polygons and G is the number of holes). All the spatial entities of 2D space-filling models (decomposition models) can be modelled within the same 2-manifold. In order for this to be possible, the geometries must conform to *planar enforcement*, a situation in which all the geometries must form a non-overlapping and exhaustive coverage within a plane. When this is appropriate for the geometry data (e.g. a map of non-overlapping administrative areas), ensuring planar enforcement is a useful validation procedure.

For space-filling representations (decomposition models), there is another reason besides validation for the importance of planar enforcement; that it allows geometry to be modelled using a compact and efficient *topologically-structured geometrical model*. Topologically-structured geometrical models describe geometry using a mixture of topological and geometrical information, usually assuming planar enforcement. An example is illustrated in figure 4.10, which is based upon ArcINFO's 'Coverage' data model. Positional information is held along chains (sequences of line segments). Each chain has exactly two bounding polygons (including the abstract 'outside polygon' of Polygon A), which is described for each chain. From the chains, the geometry of the polygons can be built. If the data being described conforms to the planar enforcement imposed by the model, this topologically-structured approach has a number of advantages over purely geometrical approaches to representation. Firstly, the geometry of boundaries between polygons is only described once and is shared between two polygons. Secondly, the stored topology is a useful property for spatial analysis. There are many variants of topologically-structured models; see Worboys and Duckham (2004); Baumgart (1972); Guibas and Stolfi (1985). Some of these store all the geometry within 0D primitives and build all the higher-dimension primitives from 0D geometry. The example shown, does not have enough information to immediately be able to build polygons from a polygon ID, so

some variants of the model give polygons a reference to one of their chains in order to facilitate this.

Non-space-filling models (constructive models) store the geometry for each spatial entity separately without any topological reference to any others. This is a simple model to implement and is used by the ESRI Shapefile file format (ESRI, 1998). Since there is no concept of topological relationships between spatial entities, no topological constraints between spatial entities are imposed. The lack of topological constraints make them suitable for all both the space-filling and non-space-filling representations. In fact, ESRI's Geodatabase data model stores all data using a non-space-filling (constructive) approach to give the widest geometrical flexibility. It allows the definition of a set of topological rules to validate the data. This means that a set of topological rules which define planar enforcement can be defined and applied to data held in a constructive model (note that this will result in storage redundancy as shared boundaries need to be represented twice). The topological rules are illustrated in figure 4.11[8].

As stated earlier, 3D b-rep models model a set of solids with a set of 2-manifolds which each comprises the boundary of the solid (one for each solid). The Euler-Poincaré formula has been generalised to validate 2-manifold boundary-representation models by introducing the concept of 'holes' and 'shells' (Hoffmann, 1989), to $V - E + F - (L - F) - 2(S - G) = 0$ (where V is the number of nodes, L is the number of edges, F is the number of polygons, G is the number of holes and S is the number of shells). This is shown in figure 4.12

As shown in figure 4.13, 2-manifold surfaces cannot be used to model two separate solids in the same manifold. Either two manifolds be used, or the geometrical model must allow for the representation of quasi-manifolds (Cohn, 1995) and non-manifolds, whose topological properties do not conform to those of 2-manifolds.

4.5.5 Dimensionality: 2D, 2.5D and 3D

In the sections above, different dimensionalities of Euclidean space, manifolds, spatial entities and dimensional element (dimensional primitives of geometrical entities) have been introduced. This section will use these concepts to try and analyse the meaning of '3D', a term about which there is often inconsistency in common use.

4.5.5.1 3D description

The key concept in terms of describing 3D geometry is whether there is enough information to place it in a 3D Euclidean metric. As seen, a 3D Euclidean space may contain different dimensional elements and spatial units within different dimensions of manifold. If these can all be interpreted and spatially related to each other, there is adequate information for a full 3D description in 3D space.

[8] http://www.esri.com/news/arcnews/summer02articles/arcgis-brings-topology.html

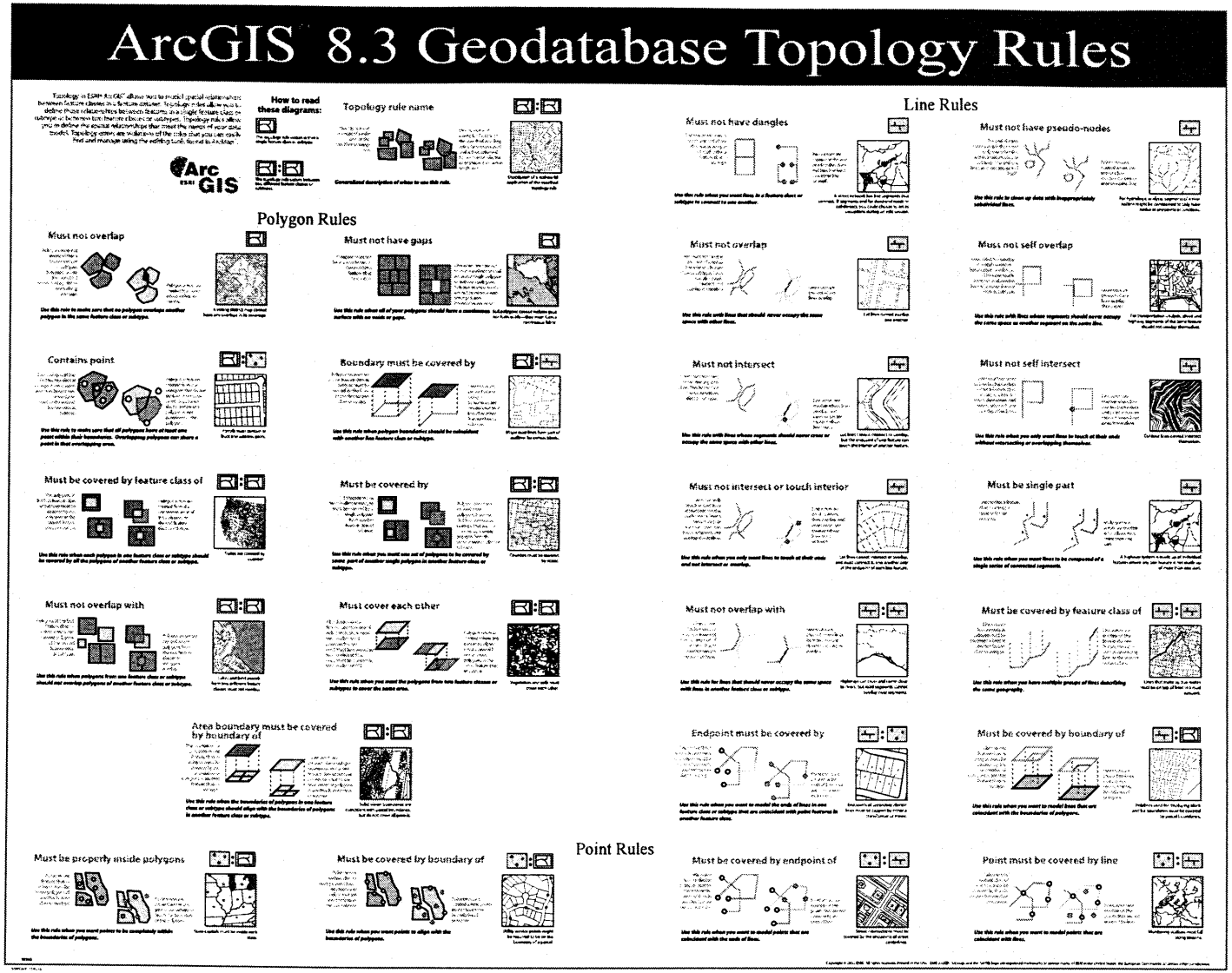


Figure 4.11: The topology rules in the ArcGIS 8.3 Geodatabase. *Source: ArcGIS 8.3 Desktop help.*

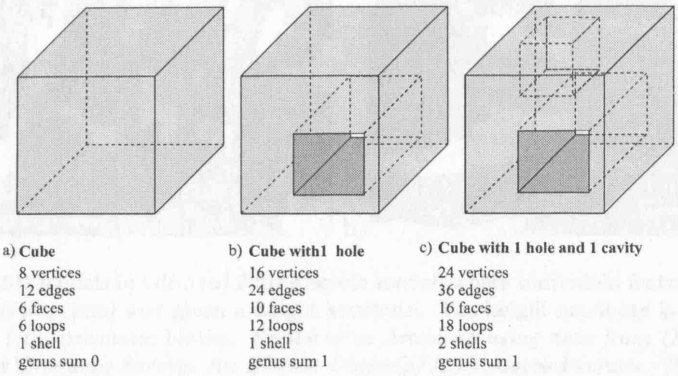


Figure 4.12: Manifolds with holes and cavities. *(a)* A cube with no holes or cavities is bounded by a set of faces (a shell), each of which is bounded by a set of edges (a loop). An edge is defined by two vertices. *(b)* A cube with a hole (a genus sum of 1). The two faces of the cube containing holes are bounded by two loops, one around the outside of the face and one around the hole. *(c)* A cube with a hole and a cavity. The cube is bounded by two surfaces (shells) (which do not touch, otherwise the form would be non-manifold); one around the outside and one around the cavity. *After Hoffmann (1989).*

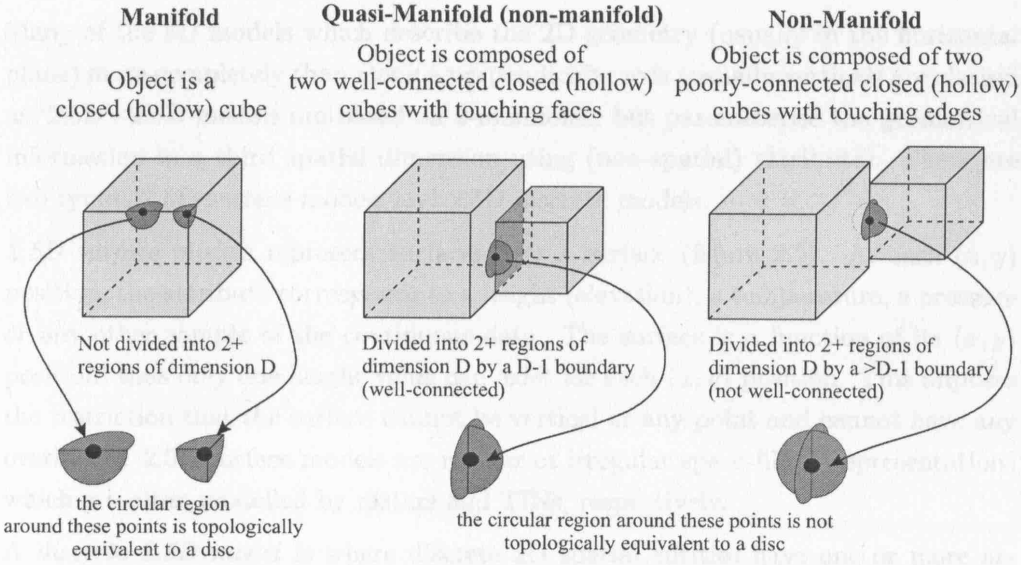


Figure 4.13: 2-manifolds, quasi-2-manifolds and non-2-Manifolds

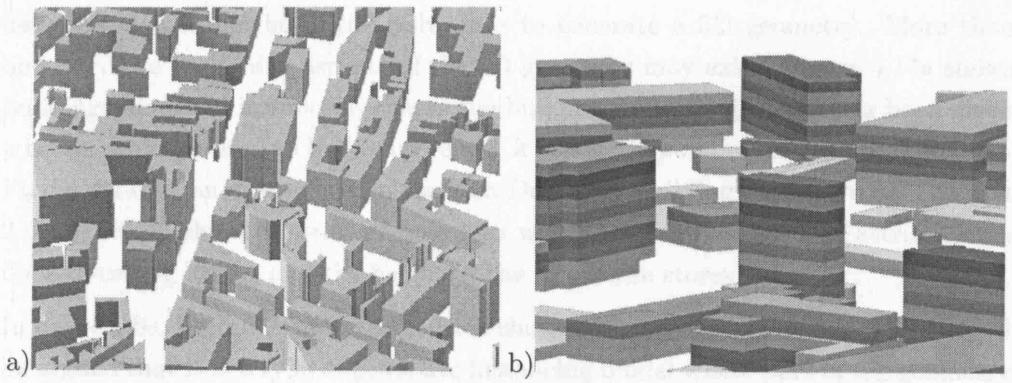


Figure 4.14: 2.5D models in GIS. *(a)* 2.5D discrete model, where individual features are modelled as 2D polygons (footprints) and given a height attribute. The height attribute is used to extrude the polygons to form prismatic blocks. *Rendered in ArcScene using data from OS and Infoterra. Crown Copyright Ordnance Survey. An EDINA Digimap/JISC supplied service. (b)* Non-Domestic Building Stock Project model using 2.5D discrete model. Individual storeys are dealt with by giving floor polygons a base-height and then extruding to the height of the storey. Overhangs, underpasses and bridges can be modelled. *Source: Non-Domestic Building Stock Project.*

The issue about whether the data is full 3D data, is to do with flexibility of representation. As reviewed, there are different models for representing geometry, each with different assumptions about the data it is to represent, resulting in different representational capabilities. Some models cannot represent curves, some rely on a fixed and limited number of high-level primitives, and some can represent the 2D geometry better that the geometry along a third axis. If the data being modelled conforms to these assumptions, the resulting description can precisely define a full 3D geometry. However, problems may arise where 3D geometry has to be simplified in order for it to be representable by the geometrical model.

Many of the 3D models which describe the 2D geometry (usually in the horizontal plane) more completely than along a perpendicular axis (usually vertical) are classed as '2.5D'. 2.5D models are based on 2-manifolds, but parameterise the geometrical information in a third spatial dimension using (non-spatial) attributes. There are two types: 2.5D surface models and 2.5D discrete models.

2.5D surface models represent surfaces over a surface (figure 2.5). At each (x, y) position, the attribute corresponds to a height (elevation), a temperature, a pressure or any other sample of the continuous data. The surface is a function of its (x, y) position; thus only one height value can exist for each (x, y) position. This imposes the restriction that the surface cannot be vertical at any point and cannot have any overhangs. 2.5D surface models are regular or irregular space-filling representations which are often modelled by rasters and TINs, respectively.

A *discrete 2.5D model* is where discrete 2D spatial entities have one or more attributes which describe some aspect of geometry related to another spatial dimension (usually along the z axis). The most common manifestation of this is where polygons are given a 'height' attribute. This is interpreted in some way – either to give the polygon a thickness (i.e. to extrude it to a simple polyhedron) or by

assigning a base height to the polygon – to generate a 3D geometry. More than one attribute describing aspects of the 3D geometry may exist. Figure 4.14a shows buildings modelled in this way where the building footprint polygon has been given a height which is used to build and render a prismatic polyhedron for each building. Figure 4.14b is an example from the Non-Domestic Building Stock Project (section 2.9.2.3) which shows a series of buildings where each storey has two attributes: a floor-to-ceiling height and the height of the top of the storey below.

In figure 4.8e, the 2.5D discrete model is shown as a 'sweep model' (though it could be argued that it is a type of primitive instancing model where part of the geometry is described using a parameter). Other aspects of the geometry can also be described using parameters; most commonly the base height. Purists would argue that this is not a 3D description because two of the spatial dimensions (x and y) are described much better than the other dimension (z). However, as argued above, since there is enough information to place it within a 3D space, it is a 3D description. The reasons for being concerned about its poor geometrical descriptive capability along the z-axis arise from a concern that an important part of the geometrical description is omitted, simply to allow the geometry to be represented. However, if the entire geometry is adequately described and there is enough information to plot it in a 3D Euclidean space, then to the end-user, the data are adequately described in 3D. This type of description has often been found to be adequate, for example, for the representation of buildings.

To provide an example, Stoter *et al.* (2002) in their work on describing 3D situations for land registries, identify three potential approaches: full 3D cadastral registration, a hybrid solution and 3D tags within the existing system. Of these, the first is the sole use of polyhedra in a 3D Euclidean metric. The second proposes a mixture of 2D registrations (polygons representing space) and 3D polyhedra, all of which are positioned in 3D Euclidean space. The final approach is the approach currently used by the Dutch land registry (figure 2.8), where a 2D system in 2D Euclidean space is used and any 3D situations are linked to external descriptions in a different modelling environments. Stoter and Ploeger (2003) identify the hybrid solution as the most promising, in part because the legal framework does not yet exist for describing all land registrations in three dimensions. However, it is made clear that this would only be in the short term because it is not a lasting, sustainable and consistent solution, and that a move to a full 3D approach is the eventual goal.

4.5.5.2 Data collection, handling and storage

3D data are more difficult to acquire than their 2D counterparts. Section 2.2 showed how laser scanning equipment can be used to acquire 3D data. However, the data are very unstructured (though algorithms are being developed to structure them automatically or semi-automatically) and it is difficult to acquire information about the interiors of buildings. Individual building plans can be used, but as explored in

Zlatanova and Prosperi (2006), integrating such data is usually a manual and very time-consuming process. Partly because of the difficulty of acquiring good 3D data, 3D models of cities tend to be rather simple (as reviewed in section 2.9.1), using simple 2.5D approaches. However, section 2.9.2.3 and figure 4.14 show that 2.5D models can be used in a more sophisticated fashion.

Another reason why 3D city models tend to be rather simple is the high storage space requirements of fully resolved 3D data; 3D city models typically extend over large geographical areas. A final reason is that many are intended for flythrough visualisations, so texture-mapped simple geometrical blocks are most suitable for the rapid visualisation and the display of apparent detail.

4.5.5.3 Ergonomics

Ergonomically, 3D data are difficult to handle (to visualise and edit) because the geometries at the front obscure those behind (depth of field). In addition, there is a reliance on planar media (paper and monitors) to view and edit data. 2D projections and slices through 3D data are commonly used for display and editing.

CAD systems and more general geometrical modelling tools have developed in ways that allow 3D data to be modelled by using 2D sections and allowing particular parts of the geometry to be isolated, rotated and edited separately.

Section 2.9.4 reviewed the use of data on buildings for the emergency services. For the manual planning of rescue or evacuation in buildings, the visualisation of storeys is usually as separate 2D floorplans. 3D perspective drawings, although perhaps useful in conjunction with the floorplans, do not assist the process much. However, digital 3D models from which floorplans could be generated would fulfil the same aim. The advantage of having a full 3D model with good framework would be that it would be possible to automate some of the planning which currently has to be undertaken manually.

4.5.5.4 Conceptualisation of volumes

It has been argued that any spatial data which can be plotted in three dimensions is 3D data; i.e. if it provides a 3D *description*. 3D spatial data may include some or all of points, lines, polygon and volumes, all of which can be embedded (plotted) in 3D Euclidean space.

As described, neither early CAD systems nor Line-Line could conceptualise polygons – they had to be built from the geometrical description. In the same way, many descriptions of 3D data do not explicitly describe volumes. Discrete 2.5D approaches describe objects with an implicit volume (which can be computed). A set of surfaces oriented in 3D space may geometrically enclose a volume, but if this volume is not explicitly represented, it has to be resolved from the purely geometrical information. The representation of explicit volumes allows volumetric analysis to be performed

and the topological relationships between volumes to be conceptualised. Early 3D CAD systems used wireframe (lines only) and surface-based geometrical models, before the conceptualisation of solids was supported.

For visualisation purposes, solids and the concept of solid 'features' may not be necessary. Examples are provided by the 3D geometrical models of some of the virtual cities illustrated in figure 2.16, in which buildings are represented by their outer shells which are cut by another surface representing the terrain.

Many GIS systems have some 3D capabilities. The ESRI Shapefile has support for 2.5D points, lines, polygons in which the vertices can be given heights, and 3D surfaces known as 'multipatches' which can be oriented in 3D space. These provide a good description of the 3D data, but there is no concept of volumes. In theory, it would be possible to write routines to do volumetric calculations on closed spaces bounded by multipatches, but such routines are not natively supported by the application.

4.5.5.5 Topology

Data may be described in a topologically-structured fashion, a structure which incorporates a topological description. In non-topologically-structured models, if topology is needed (for validation or for spatial analysis), it needs to be built. Some systems will compute the topology and store it; others will build it when required (Breunig and Zlatanova, 2006) which takes more time but less storage space.

Although GIS software packages are capable of describing points, lines, polygons and surfaces in a 3D Euclidean metric, they do not represent volumes, and crucially, they only deal with 2D topology. This is often the criterion used to distinguish between 2D and 3D analysis tools. If 3D topology is not supported, full 3D analysis is not possible (e.g. adjacency relationships above and below).

4.5.5.6 Discussion

Dimensionality applies to data description, completeness of representation and topology. A full 3D modelling system should be fully 3D in all these three aspects. However, any spatial data which can be plotted in a 3D Euclidean space (including 2.5D approaches) can be imported into a full 3D system for its topology to be built and spatial analysis to be performed. For this reason, frameworks for 3D data description and frameworks for 3D data handing and analysis are distinguished.

Section 3.4 asserted that the framework being designed here should act as a data repository; a description of data. Since it begins with 2D data and is enriched with 3D data, it should be able to cope with both 2D and 3D descriptions, and thus be able to cope with incompleteness in data description along the z axis.

4.6 Features

The term 'feature' (when used alone in the modelling context) is a conceptualisation of a real world feature – an instance or realisation of a concept and Kottman (1999) describes the importance of *persistent identifiers* for identifying them. In the context of geospatial data, the feature is normally considered to be geometrical and is defined by the Open Geospatial Consortium as the "fundamental unit of geospatial information" which "corresponds to a real-world or abstract entity" (Kottman, 1999). However here, two types of feature are considered; geometrical features and non-geometrical features.

As is apparent from the examples in chapter 2, there is an infinite number and manner of ways in which to conceptualise the real world as a set of discrete and distinct features. Section 4.6.1 shows the issues and difficulties associated with defining features. The 'conceptualisations of real-world features' design principle (section 3.1) distinctly calls for the framework to support multiple conceptualisations of features. The National Land and Property Gazetteer (NLPG; section 2.9.2.1) allows different types of features to be defined, as described in BS7666 (section 4.6.5 and table 4.1).

4.6.1 Defining and classifying features, using 'buildings' as an example

Buildings can be conceptualised according to physical, historical, functional and legal considerations.

Bennett (2003, 2005) explores aspects of the classification of buildings and other features in the physical environment. He identifies *physical*, *historical*, *functional* and *conventional* properties as those which can be used for defining and classifying building features. Here, the further property of *access* has also been added. These will be discussed below.

4.6.1.1 Physical properties

Physical properties are the most obvious and are likely to be the properties in most widespread use. Both Ordnance Survey (shown in section 2.4.2) and the 'Four Towns Database' (Holtier *et al.*, 2000) use physical properties to define building units (geometry and building material). For the Four Towns Database, the reasons were clear – both are likely to affect the energy use of the building (heat loss from the building and heat gain from the sun). For Ordnance Survey, the outward physical appearance of buildings is used as an indication for its division into functional units.

The classification scheme of Steadman *et al.* (2000) is based on the geometrical properties of buildings as supplied by the Four Towns Database. Minor appendages such as balconies are treated separately, and geometrically complicated buildings

are broken down into smaller component parts whose shape and dimensions are described. The component parts are then classified according to geometrical form and the character of the arrangement of spaces. Finally, composites of these parts are classified. This scheme was developed for the analysis of energy efficiency in buildings.

Not all physical properties of buildings are sharply defined. Areas of roof which cover open space not bounded by walls, do not create completely bounded volumes of space. Examples of these types of situation are roofs attached to the sides of buildings and open-sided covered walkways. The covered space is more open to being affected by wind and rain, particularly if it is a strip of space close to an open-sided edge of the roof; the extent of the affected area is affected by its aspect, the height of the roof and the local effects of the surrounding buildings and trees.

4.6.1.2 Historical properties

A building's origin and the reason it was constructed is an example of a historical property. The building may have been built as part of, for example, a council house construction programme or in order to exploit a large amount of land which has just become available for redevelopment. It is likely that the circumstances under which buildings were constructed give rise to particular properties.

Buildings frequently undergo refitting, modification or change in use. The history of such changes may be correlated to wider events, changes in fashion, changes in legislation, or may just be due to events affecting the building. The history of change in a building could be used to assist in classify buildings. In terms of defining the extent of feature, buildings could be subdivided into parts based their history of use, modification and construction (e.g. an extension to a building).

Periodic changes in use also occur, such as a weekly change of function of a church hall into a soup kitchen.

Bennett (2005) notes that it is rare that much information exists on the history of features.

4.6.1.3 Functional properties

Lawrence (1989) discusses how humans partition space, how they classify and name these partitioned spaces in and around buildings, and how they relate these spaces to each other using relative terms such as 'inside/outside' and 'upstairs/downstairs'. He discusses how the relationships between the design, meaning and use of space in the built environment, can be described in terms of classifying the activities which take place, describing the relative positions of activities, comparing the relative positions of different activities, and how comparing the different activities are grouped and related to each other. For example, in a domestic house, cooking and eating may take place in the same room, whereas in a restaurant, they take place in different

parts of the building. In a shop, the main activity (selling) usually takes place at the front of the shop, whereas in a theatre, the front is associated with selling tickets and waiting, whereas the main activity takes place further inside.

Lawrence (1989) also explores notions of inside/outside and public/private as means for demarcating space. This was an issue which also interested Nolli in his map of Rome (section 2.9.3.1) and more recently, Hwang and Koile (2005) who are investigating a machine-learning approach to identifying public and private space based on fieldwork to survey the public perception of these distinctions.

Bennett (2005) considers three types of classifying the function of buildings: potential function (of which there is likely to be a huge variety), actual function and intended function. In addition, functions change with time.

Ordnance Survey's feature catalogue defines a hierarchy of building types (Ordnance Survey, 2001); for example the 'Place of Worship' class which includes 'chapel', 'synagogue', 'temple', 'abbey' and 'burial ground'. Bruhns *et al.* (2000) review classification schemes for buildings as originally used by the Four Towns Database, the UK Valuation Office and the schemes based on the Standard Industrial Classification scheme.

4.6.1.4 Access properties

There is plenty of evidence to suggest that the properties of access between spaces may be useful for grouping together (thus defining a geometrical extent) and classifying spaces. From a topological analysis of five residential building plans, Lawrence (1989) found a 'privacy gradient' from the most accessible entrance points and hallways to the least accessible private rooms. He found a correlation between the relative privacy (accessibility or depth) of spaces and room type (which correlates to function and activity), but found that the details of the correlation varied depending on the part of the world being studied; in his Swiss examples, the parents' bedroom was the deepest space (most private), whereas his earlier study in Australia found that the parents' bedroom was commonly the shallowest (least private) space and the space in which guests to the house were invited to leave their coats. Other examples of the correlation between the depth of space and the functions of rooms in the context of the building are discussed in chapter 4 of Hillier and Hanson (1984) and some other examples are illustrated in figures 4.18 and 4.17.

Space syntax (Hillier and Hanson, 1984) is a method of analysing the structure of space, which has been applied both to inside buildings and also to the structure of urban space. Space is represented by a set of straight axial lines which are fitted within convex space to produce a network-like structure. Axial lines are often seen as being lines of sight, although they do not generally take account of the fact that sight diminishes over large distances and do not take into account the obscuring effects of an undulating terrain. These types of problems are subjects for research;

for example, Batty and Rana (2002) describe a method for automatically defining axial lines based on viewshed models. The number of axial line intersections for a single axial line is often used as an indication of accessibility (for the convex space represented by the axial line) and axial graphs are transformations of axial maps in which the connectivity of spaces is depicted. A variety of statistics can be generated; an example of one is *integration* which is a measure relative depth of spaces from each other, a similar concept to that used by Lawrence (1989).

Consider a terrace row of buildings defined by outer appearance as in figure 2.11: the functional and access separation between the individual terraced units can be inferred. However, the presence of a connecting door would change the accessibility relationships and perhaps the functional relationships. A connecting door may be used for infrequent access, in which case the functions of the two building units would not be much changed. However, if the door was used frequently or especially if parts of the party wall were removed, the functional pattern of the building might drastically change. It is reasonably common practice to replace the internal structure of a building in order to preserve its outer appearance, which results in a complete change of accessibility pattern and function (e.g. conversion of old warehouses to residential flats).

4.6.1.5 Properties associated with convention

Properties associated with convention are those which do not necessarily fit into those described above; they are seemingly rather arbitrarily defined – by some convention.

Legal definitions, such as those held by land registry organisations, are included as a property of convention. Although in practical terms these *tend* to coincide with physically, historically, functionally and accessibility defined boundaries, they do not necessarily need to. Bennett (2005) notes, for example, that the ownership of parts of open space (gardens, yards and pavements) can appear to be arbitrarily defined and thus cannot be inferred from any of the other properties.

The definition of postal delivery points is another example of a property which *tends* to correspond to physical, historical or functional characteristics of the built environment, but not necessarily. Postal delivery points are cannot defined by a fixed set of criteria related to the other properties of the built environment (Yildirim and Yomralioglu, 2004). Some postal addresses are not registered (such as basement flats) and only have post delivered because of the local knowledge of the postal deliverer. Some groups of buildings (residential flats or offices belonging to or managed by a single organisation) have one delivery point from where post is subsequently redistributed internally. The mismatch of address and properties is a major problem and this is one of the reasons for the BS7666 standard (section 4.6.5) and the NLPG (section 2.9.2.1) initiative.

4.6.2 The geometry of geometrical features

As illustrated by these examples of how buildings can be defined, a feature's geometry is inextricably linked to its definition. Once a feature has been conceptualised, its geometry can usually be surveyed and a snapshot of its geometrical state recorded.

4.6.3 Attributes of features

Both geometrical and non-geometrical features have attributes (properties). These usually take the form of a set of pairs of fields and values which may be represented as a table; this is the case in most GIS software packages. The attributes and possible range of values chosen are strongly dependent on the application domain and conceptual model. As for feature geometry, attributes of features may change through time.

4.6.4 Classification of features

Natural language words and expressions are in common use to classify real-world features (e.g. 'house', 'garden', 'station', 'lake', 'river'). The common understanding of these terms is not rigorously defined but is suitable for everyday conversation. Formally, the use of such words is problematic without a well-defined ontological[9] model of concepts and the semantic mapping of these terms onto these concepts. The consideration of ontology and associated semantic mapping is an important component of building frameworks for computer based interpretation and data interoperability (Bittner *et al.*, 2006).

The way in which the semantic term 'church' is defined may be with reference to any of the properties discussed; i.e. it may refer to buildings which have the physical appearance of churches (e.g. with steeples), which were built to be churches, function as churches or are defined legally (e.g. as workplaces of a priest or vicar). Historical criteria are clearly time-dependent, but functional aspects of buildings can likewise be time-dependent. Church or community halls are often host to multiple activities at different times of the day or the week (e.g. plays, sports, canteens and fairs). Thus, a building may change its classification at different times or may be defined according to the 'main' activity.

Steadman *et al.* (2000) describe a physically-based classification scheme designed to assist in energy analysis in buildings. The scheme is based on a classification of the geometrical forms of buildings, a classification of the arrangement of spaces (individual rooms, hall or open-plan) and whether space is naturally or artificially lit. Classifying buildings in these terms can partition them into groups useful for studying energy efficiency. This scheme was developed using empirical data from

[9] 'Ontology' concerns the meaning of concepts of the feature, their relationships to other concepts and the mapping of semantic terms to these concepts.

Table 4.1: Allowable BLPU provenance codes. *Source: British Standards Institute (2000b, table 5, p7)*

BLPU provenance	BLPU provenance description
Registered land title	Title registered by HM Land Registry or Registrars of Scotland
Unregistered land title	Title deeds or similar (but not registered)
Formal tenancy agreement	Subject of formal tenancy agreement
Rental agreement	Subject of rental agreement
Physical features	Defined by physical extent
Occupancy	Defined by identified occupancy
Use	Defined by established use

the surveys of the four towns made in the NDBS project (section 2.9.2.3). As stated, the data provider usually uses a scheme to fix a classification and identity to each object.

An alternative approach is to define rules for the automated classification of features based on other aspects of features such as their geometry (e.g. ratios of dimensions), their attributes, those of surrounding features and perhaps the wider spatial configuration of the surrounding features. This would allow data users to apply their own restricted ontological model for their features; restricted because, clearly, the range of supported ontological models is directly dependent on the choice of attributes given to features and the way in which the attributes are expressed. A simple example could be to define a 'skyscraper' as a building having over ten storeys and to use this rule to automatically classify buildings.

The discussion above assumes that features are at a fixed scale and are discrete. As the scale of representation changes, classification schemes change. For example, at a city-wide scale, it is likely that a classification scheme would distinguish between different types of building, whereas at a nationwide scale, it is likely that aggregations of buildings would be of interest, rather than individual buildings and their individual types. In the natural environment, rivers, streams and lakes cannot necessarily be discretised with any certainty. The point at which a stream becomes a river, a river becomes an estuary and wider portions of a river become lakes, is scale and domain-dependent. Rules for classification could be scale-dependent, involving geometrical constraints and the speed of flow. Thus, there could be multiple sets of classified discrete features, which would be scale-dependent and possibly seasonally dependent.

4.6.5 Case study: BS7666

The British Standard BS7666 is the standard behind NLPG (section 2.9.2.1) which is in four parts (British Standards Institute, 2000a,b,c,d): street gazetteer, land and property gazetteer, addresses and public rights of way. It was created to facilitate a national land and property gazetteer. Figure 4.15 shows the logical model of the

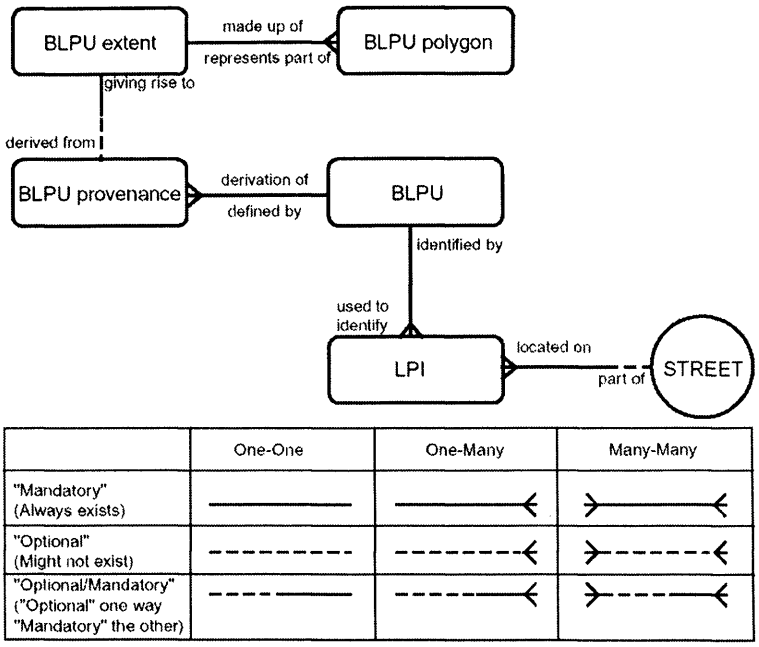


Figure 4.15: Logical data model for a land and property gazetteer from BS7666. *Source: British Standards Institute (2000b, p4)*

gazetteer. The central concept is the 'basic land and property unit (BLPU)' which is defined as an area of land, property or structure of fixed location having uniform occupation, ownership or function (British Standards Institute, 2000b, p1). Although not shown in figure 4.15, a BLPU can be a ' primary addressable object', otherwise it is a 'secondary addressable object' if it forms part of a primary addressable object. Each BLPU is uniquely identified by a 'unique property reference number (UPRN)' and always has one or more 'land and property identifiers (LPIs)' and one or more 'BLPU provenances'. Each LPI is a piece of human-readable text, usually the street address or addresses where multiple valid addresses exist. The BLPU provenance is the classification of what the BLPU represents; see table 4.1 for the current possibilities defined by BS7666:2. Each provenance has an optional extent associated which is made up of one or more polygons (all BLPUs have a representative coordinate, regardless of this). This structure is flexible and powerful, because is allows *one BLPU concept* to have more than one *aspect* about it recorded, each of which may have its own extent, as defined by one or more polygons.

For example, consider a block of flats surrounded by garden. This concept would be represented by a BLPU (with a UPRN and coordinate), identified by a LPI representing its street address. One provenance would be a registered title and the extent would be a polygon encompassing the block and its garden. A number of garages at the edge of the garden might be rented out separately, so a number of other provenances would be classified as 'subject of rental agreement' whose extents were represented by polygons representing the garages' footprints. The block might contain eight flats, so there might be eight secondary BLPUs associated with the pri-

mary BLPU, each with one or more provenances. It is clear that complex situations can be described with BS7666:2.

4.7 Access

As shown in section 2.9.3, access is a very important aspect of the built environment. Being designed to house human activity, the built environment relies on human access in order to host the activity. There are three aspects to consider, when considering access:

- Representation of the state of access within the environment
- Measures of accessibility
- Prediction of movement and behaviour

The subject of the predictive modelling of movement and behaviour in itself is outside the scope of this thesis. However, it is an important potential application involving access and like all models of *process*, representations of the starting, intermediate and projected *states* of the environment are required.

4.7.1 Representation of the state of access

Representations of *state* are essential to any model of *process* (section 2.1); for example, representation of the state of access is essential for predictive models of pedestrian behaviour.

In the purest terms, access can be considered as a topological relationship. Given a set of discrete locations or spaces, access can be considered to be the topological connections between pairs of those locations or spaces. Such relationships can be described using a graph consisting of a set of vertices connected by edges (lines) which represent relationships between the vertices which they connect. In these *access graphs*, vertices represent discrete locations or spaces, and edges represent the presence of direct access between pairs of locations or spaces Steadman (1983, p75). Ordnance Survey's 'Integrated Transport Network (ITN)' (section 2.9.3.2) is a example of such an access graph.

A pure topological conceptualisation of access is rather limited, because access is not binary. There are different qualities and types of access relationship and different costs associated with them. An example is distance – the cost associated when accessing a location which is far away is likely to be more than accessing a closer location. Such costs are not always bidirectional; the gradient of the terrain over which a route runs will affect the cost in different directions; i.e. whether the agent is moving with, against or oblique to the gradient. Such information can be embedded into graphs as attributes on edges and vertices (Tinkler, 1977). ITN's 'Road Routing Information (RRI)' (figure 2.21) attaches time- and vehicle-dependent information

to the links and vertices of ITN which may affect access. In the case of RRI, a binary decision on whether access is allowed can be dependent on the day, time of day and the type of vehicle. Geometrical information on the gradient and the maximum width and height of any vehicle which is able to pass, is included

The geometry of 3D Euclidean space (or at least 2D space) is potentially of great importance for access. The acuteness of bends in the road network, the gradients and the widths of roads, and the headroom, all affect the types of vehicle which can use them and the speeds at which they can travel. Pure topological networks do not depict geometry; i.e. the positions of vertices and the angle and linear geometry of links do not necessarily correspond to the Euclidean geometry of the network. Geometry can be completely parameterised by adding distance and width attributes to edges. *Geometrical networks* are those which incorporate both topological and some geometrical information. Here, the positions of vertices have a geometrical correspondence with actual positions in 2D (or 3D) Euclidean space and edge serve not only as topological connectors but also the 1D geometry embedded in the 2D (or 3D) space. Since edges are 1D, these geometrical networks cannot depict the width and height of spaces through which the routes pass; in order to represent these, they need to either be parameterised, or be linked to external geometrical elements. ITN is a geometrical network in 2D space. Lee (2004b), however, uses geometrical networks in 3D space for describing access across multiple storeys in buildings, where edges correspond to the centrelines of the floors of corridors.

From an application point of view, as illustrated by the case study of the inappropriate mass routing of vehicles through a small village, there is always the dilemma of whether to include informal routes or 'appropriate' routes in route suggestions – routes which may only be known to a few locals, or routes which are strictly not supposed to be used such as short-cuts through privately owned car parks. Some informal routes might be welcomed; for example, a shortcut through a shop may result in more purchases, balancing favourably with the inconvenience of potential overcrowding.

4.7.1.1 Pedestrian access

Most solutions to describe pedestrian access use a geometrical network-based approach. ITN does this in 2D space. Lee (2004b) uses geometrical networks in 3D space (figure 4.16), automatically generating centrelines of spaces from the full geometry of buildings, after which the geometrical extent of the spaces are discarded. His network does not incorporate any RRI-type information. This geometrical-network-based approach is appropriate for routing pedestrians over large distances, but the reliance on representing access as linear routes through spaces (as graph edges) rather than through open space, is simplistic for routing or modelling the microscale movement of pedestrians, particularly where there is interaction between pedestrians.

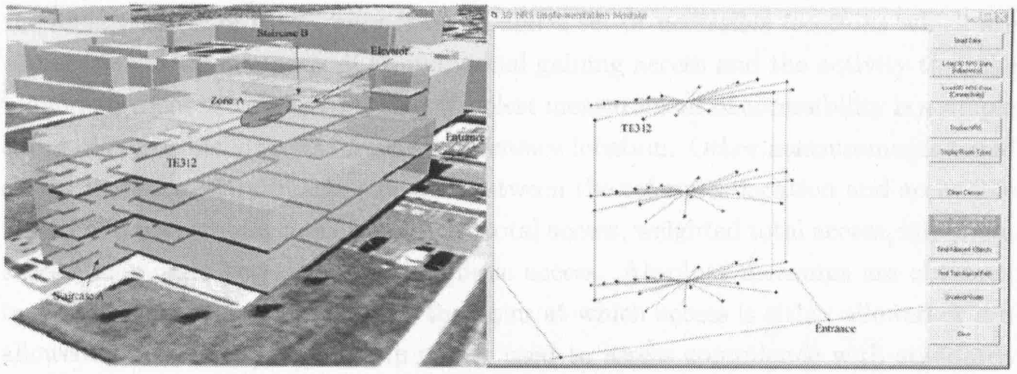


Figure 4.16: Geometrical networks in 3D space, describing pedestrian access on multiple levels of buildings. *Source: (Lee, 2004b).*

There are two problems with using such a network-based approach.

The first problem is that networks either do not represent any geometry (in the case of pure topological graphs) or only represent geometry as fixed linear routes. In the case of purely topological networks, network edges (routes) can either be attached to external geometrical descriptions of the spaces through which the routes run, or some of the geometry of the space can be parameterised. In the case of geometrical networks, open spaces cannot be represented; though Okunuki *et al.* (1999) represent open spaces using a highly-connected network mesh, completely covering the open space in a network grid. This vastly increases the search space to network-based routing algorithms and pedestrians are still restricted to fixed routes, albeit, a very large number of them.

The solution which is developed in section 5.7 is to embed RRI-type information into the *geometry* of spaces in the built environment rather than embedding it into an abstraction of the geometrical networks.

4.7.2 Aggregate measurements of accessibility

Access can be inferred from floorplans by humans as long as there is sufficient richness of symbology and annotation. For the automated assessment of accessibility, a representation of access and the environment is required, underpinned by an explicit framework.

Marshall (2005) develops methods for analysing street patterns and street layouts quantitatively. Although not his primary aim, he hopes that such measures may assist in the understanding of what constitutes good design, by applying these quantitative measures to case studies. It is for similar reasons that quantitative measures of accessibility are important; for example, for assessing the value of measures taken to facilitate better pedestrian access.

Church and Marston (2003) review approaches to measuring access, specific to a particular activity (e.g. shopping, going to the doctor). These measures rely on

the identification of a *reference location* and a set of *accessible locations* which are usually chosen on the basis of an individual gaining access and the activity that the individual wishes to undertake. The simplest measurement of accessibility is a simple count of accessible locations from the reference location. Other measurements based on the distance, attractiveness or cost between the reference location and accessible locations (or a combination of these) are total access, weighted total access, minimum access, maximum access and probabilistic access. Absolute measures are obtained by applying a threshold indicating the point at which access is either allowed or not allowed. These measures are commonly used to assess compliance with standards and requirements for disabled access, but are unable to quantify the potential extra cost to the individual of using the facilities; e.g. the gradient of a ramp affects the ease of use.

Church and Marston (2003) develop multiple activity accessibility measures (for an individual who undertakes a set of activities) based on the above, but combining accessibility measures for the different activities, weighted by their importance. The resulting *weighted gross accessibility* estimates total accessibility from a location according to a *certain type of access.* Examples of types of access are frequent short trips to a small number of locations, commuting and exploring. These types of access are likely to be dependent on the type of individual (age, ability, level of activity). Perhaps there may be activity subsets such as weekend activities, holiday activities and a working day. A more probabilistic statistic can be defined by weighting the accessibility of each component activity according to the probability that the activity will be undertaken.

The multiple activity measures from the previous paragraph are dependent on the set and weight of the activities chosen, which in turn are affected by the particular type of individual. Church and Marston (2003, cited in Golledge, 1994) discusses how the same geographical space can have very different accessibility implications for different groups of people with different mobilities. Wheelchair users cannot negotiate steps so are required to take alternative routes if available, which may be significantly different from other routes. With this in mind, Church and Marston (2003) introduce the concept of *relative accessibility* which compares the relative accessibility between different groups of people. It can also be used to assess the relative impacts of positioning ramps in buildings on accessibility for groups of people who depend on the ramps.

The measures reviewed are global measurements based on the accessibility from one reference location to a subset of all possible locations and are dependent on the reference locations chosen. Sakkas and Pérez (2005) apply similar accessibility measures to pedestrian movement in buildings for which it is necessary to quantify multiple access routes between multiple positions. Key to their approach is the identification of the *representative* positions, and routes between these positions, which *are able to service the activities associated with the building's function.* These activities are classified as *core services* (activities consistent with the building's main

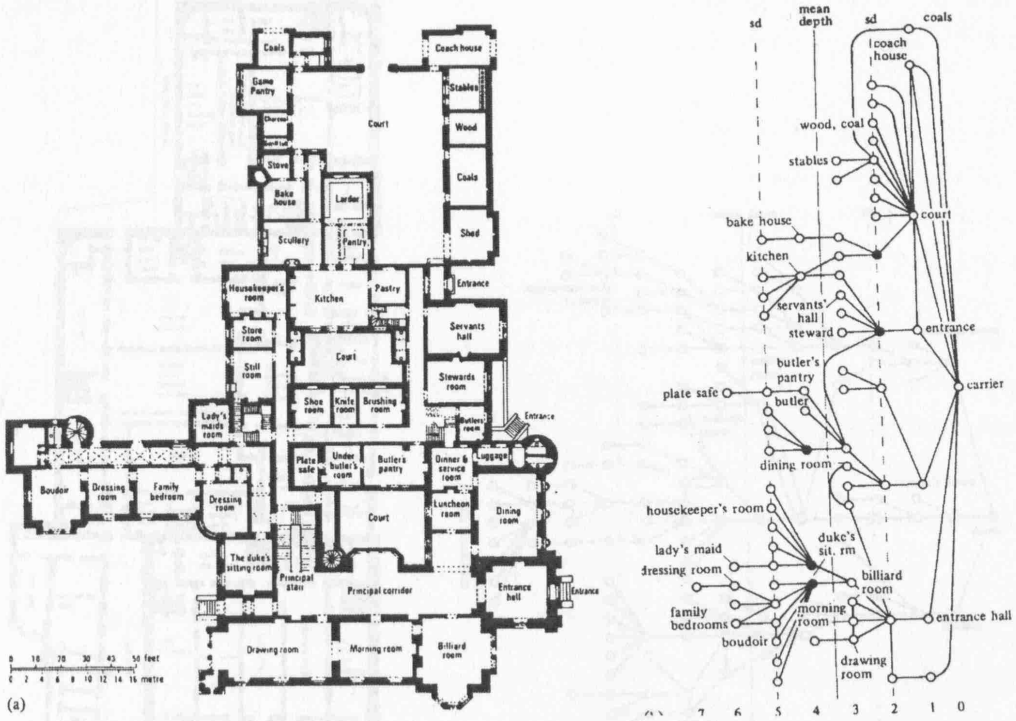


Figure 4.17: Floorplan and justified access graph of Buchanan House, Stirlingshire (1851-1853), showing that the owners' private rooms are in a deep part of the graph, beyond the spaces where visitors are entertained. Hallways are well-connected. Servants' spaces are shallower, the deepest place being the plate safe. *Source: Steadman (1983, p230); the floorplan was originally from Girouard (1971, p23) and the graph is based on Hillier et al. (1978, p25).*

function) and *auxiliary services* (other often essential activities such as fire safety measures, evacuation measures, car parking, preparation of hot drinks, and toilets). Positions include entry points outside the building in order to service access to the building by foot, by car and by public transport; this is because accessibility to the building itself is likely to be as important as internal access.

Steadman (1983) shows how graph theory can be used to study the properties of accessibility. The connectedness of graphs (ratio of vertices to edges), the longest and shortest paths, depth and breadth are examples of gross quantitative descriptions of graphs. Marshall (2005, ch5) also develops approaches for quantifying various aspects of network structure. Figures 4.17 and 4.18 show how the structure of the access graphs of buildings can be correlated to building function. The figures show single floorplans accompanied by justified access graphs. The implications of slight adjustments of the graph can be seen, such as connecting or disconnecting two poorly connected subgraph components.

Access patterns in cities also indicate their character. For example, some parts of cities are built for private cars, some for public transport and some for pedestrians. The proximity of residential housing to retail or service sectors is important because this may indicate whether walking or driving are the dominant forms of travel; this is also affected by the proximity of out-or-town retail parks and the provision

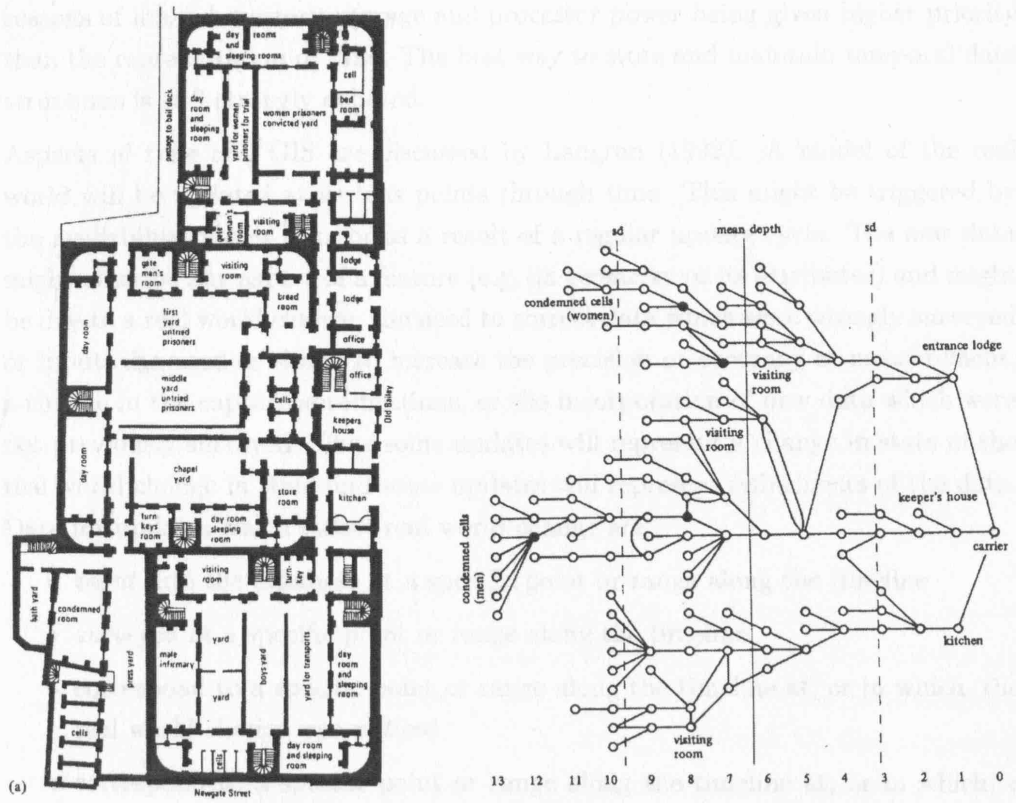


Figure 4.18: Floorplan and justified access graph of Newgate Goal, London (prior to the c1836 alterations), showing that access to most of the building is tightly controlled (most space is poorly-connected deep space). Condemned prisoners are in deep poorly-connected parts of the graph and visitors are allowed into visitor rooms which were easier to supervise about half way into the depth of the graph. *Source: Steadman (1983, p230); the graph is based on Hillier et al. (1978, p72).*

of transport to these. Parts of cities with bustling high streets which are within walking distance of residential housing will have a very different character from purely residential areas. Talen (2002) suggests that the access characteristics of cities can be used to indicate their urban quality. This is to do with factors such as the mixture of land uses, whether cities are built for pedestrian, public transport or private cars and the provision of public parks.

4.8 Time

Time is a difficult concept to deal with in databases and models of the real world. Due to the nature and scope of the conceptual model being designed, the concept of time will be conceptualised as a linear timeline. (Other methods of conceptualising time exist in the philosophical and psychological literature).

Research in temporal databases has taken place independently from spatial databases (Sellis, 1999) and has been traditionally neglected in GIS. One reason for this is the difficulty of conceptualising and formalising the rather abstract concept of time. Another reason is technological constraints; Worboys and Duckham (2004) cite the

reasons of limited memory, storage and processor power being given higher priority than the representation of time. The best way to store and maintain temporal data structures is still strongly debated.

Aspects of time and GIS are discussed by Langran (1992). A model of the real world will be updated at various points through time. This might be triggered by the availability of new data or as a result of a regular update cycle. The new data might describe any aspect of a feature (e.g. its geometry or its attributes) and might be due to a real world change, the need to correct data which were wrongly surveyed or input, the need or desire to increase the precision or accuracy of measurement, a change in the capture specifications, or the incorporation of new data which were not previously surveyed. Thus some updates will represent a change in state of the real world change in state and some updates will represent refinements of the data. Data for updates which reflect real world change are:

- *input* into the database at a specific point or range along the timeline
- *surveyed* at a specific point or range along the timeline
- correspond to a specific point or range along the timeline at, or in which, the real world change was *noticed*
- correspond to a specific point or range along the timeline at, or in which, *a proxy for the real world change* occurred e.g. a flood
- correspond to a specific point or range along the timeline at, or in which, the data were *input into the database*; 'database time' (Langran, 1992, p34)
- correspond to a specific point or range along the timeline at, or in which, the real world change *occurred*; 'world time' (Langran, 1992, p34)

This acknowledges that the ideal case where real world changes in state are reflected in the database is unlikely be met, and provides a way of managing this.

Such updates assume instantaneous changes of state. Some real-world changes are gradual so the instantaneous events assumed above do not apply (Allen, 1983). 'Wear and tear' (such as the wearing out of a road surface) is a processes which lasts the lifetime of the object, but the rate of which is variable, being dependent on other events. Changes in state which may affect other processes occurring in parallel. Just as the modelling of continuous space is usually done by discretising it into spatial units (section 4.5.2), the same is true when representing continuous time. Updates to reflect these changes might be according to some threshold of deviation between the current state and the last recorded state, or might be measured in discrete time steps. Finite-difference models use discrete time steps; for example, a cell-based river system model discretises space into cells, and time into timesteps, computing water flow and quantity between discrete spatial units and time steps. Section 2.9.3.5 discusses the implications of modelling access in a dynamic world whose state changes over time.

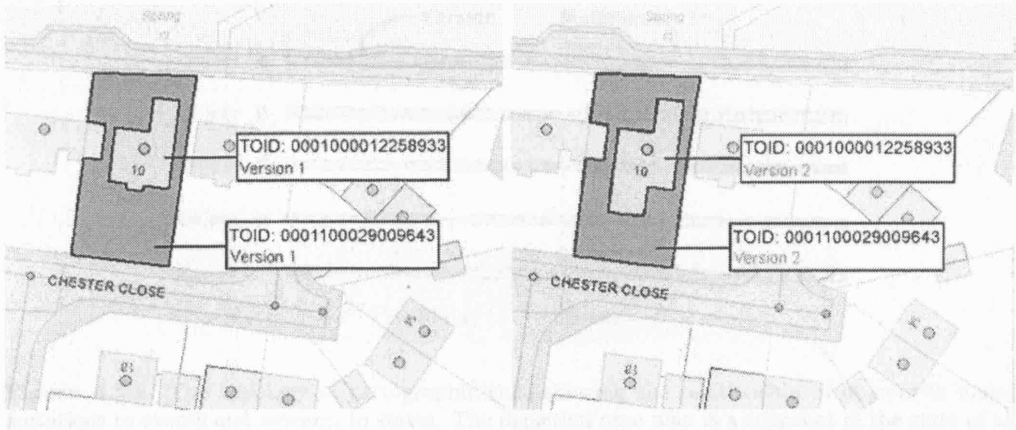


Figure 4.19: Illustration of how the geometry of a feature can change in time. In OS MasterMap, an extension being built on a house results in the same feature having a new version number and a new geometry. *Source: http://www.ordnancesurvey.co.uk/oswebsite/products/osmastermap/faq/general/index.html*

The practical implication of these issues generally only applies where full temporal database is sought, where the ability exists to retrieve past states over the continuous period for which the modelled data is valid. Where full temporal database is not required, superceded data are simply removed, because all that matters is current state being as up-to-date as possible. If a full temporal database is sought, distinguishing between these types of time and the whole update history is vitally important. A period of poor-quality data input will clearly affect the quality of data retrieved for that period.

Since the database is likely to contain snapshot states, some method of *interpolating a state* between two snapshots is required. If instantaneous changes are assumed with a steady state in between (this is likely to be the case), this is reasonably straightforward – the interpolated state is the same as the earlier state. If, however the process is known to be gradual this is more difficult and a good knowledge of the mechanics of the process is required. If a distinction is made between the respective times at which the state occurred, was noticed, was surveyed and was input, it should be possible to retrieve the alternative realities of the real world state, the state of knowledge, the surveyed state and the state of the database for any instant along a timeline.

In feature-based representations, a feature has an *identity*. The *state* of the feature can be changed by an event. Often a feature whose state has changed retains the same identity. Bittner *et al.* (2006) provide the example that although cells in a living organism are being constantly replaced, the identity of the organism remains. Similar examples can be found for the urban environment; e.g. the partial rebuilding of a building after fire damage or the construction of an extension to a building. Figure 4.19 illustrates a change in a feature in which it retains its identity. Note that although the change has not caused a change in the feature's identity, it is allocated a new version number; the feature's history of change becomes part of

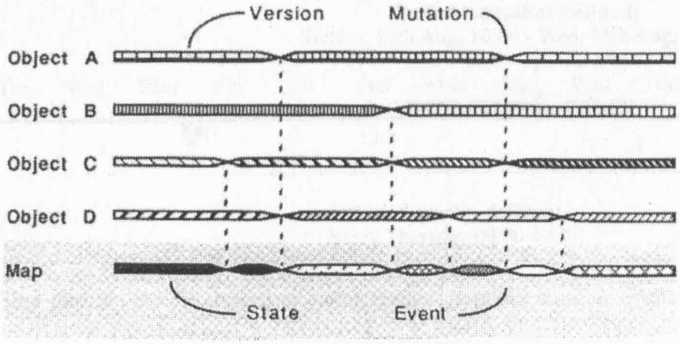


Figure 4.20: The topology of cartographic time showing the relationship of objects to maps, mutations to events and versions to states. The depiction on a map is a snapshot of the state of all objects without reference to the events. *Source: Langran (1992, p33).*

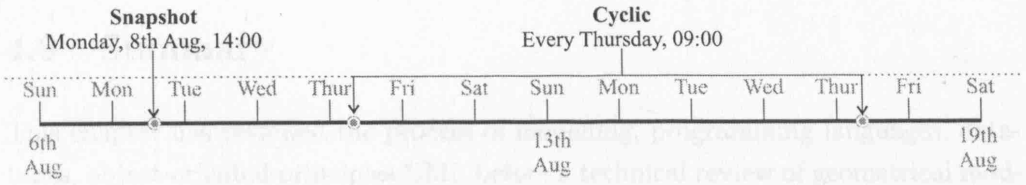


Figure 4.21: The conceptual model distinguishes between two types of time: *a)* Snapshot time; and *b)* Cyclic time.

the feature's identity (Bittner *et al.*, 2006). One can envisage each feature having a timeline punctuated by instantaneous events, which cause 'mutations' (changes in state) of the feature (figure 4.20). Between instantaneous events along the timeline, features have states which are temporally bounded by events.

There are cases in which the feature is so fundamentally changed that it ceases to have its original identity, and assumes a new identity. The complete demolition and rebuilding of a building may, in some cases, be considered to create a new building. A more complicated example is if a semi-detached building represented as two contiguous features has its dividing wall removed, so that either the new combined feature assumes a new identity or it assumes the identity of one of the original features. The amount and type of change necessary for a feature to assume a new identity is not obvious and can be considered to be application specific. Ordnance Survey has (mainly geometry-based) rules for defining the amount and type of change which cause one of its features to change its identity and be allocated a new TOID (section 2.3.5.2).

4.8.1 Snapshot time and recurring (cyclic) time

Figure 4.21 shows that two types of instantaneous time can be distinguished, *snapshot time* and *cyclic time*. Snapshot time is a one-off point in time and is appropriate for describing the time at which a feature was surveyed, for example. Cyclic time recurs in a repetitive fashion; examples of which are daily, weekly and seasonally.

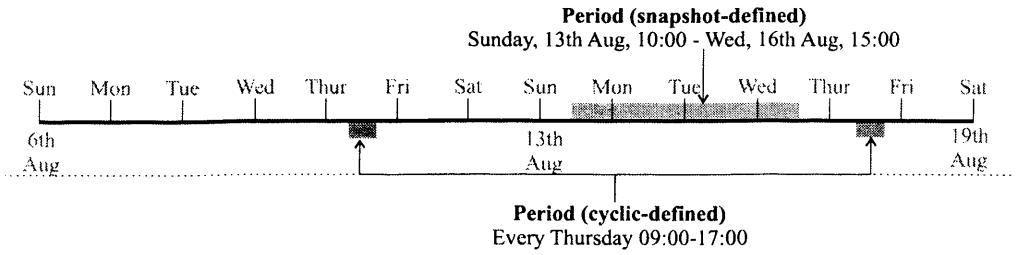


Figure 4.22: Time periods can be described using either snapshot time or cyclic time.

Figure 4.22 shows the equivalents when applied to time periods rather than instantaneous time snapshots.

4.9 Summary

This chapter has reviewed the process of modelling, programming languages, databases, object-oriented principles UML before a technical review of geometrical modelling, features and the representation of access.

This theory will be extensively drawn on in the subsequent two chapters.

Chapter 5

The Data Model

This chapter is a system-independent description of the data model which addresses the design issues of chapter 3. Since these design issues are within the remit of both the conceptual and logical models, this chapter comprises not only the conceptual model, but also the strategy for implementation, including some specific storage details. Well-defined concepts are identified, which are formalised into *entities* whose roles and the relationships between them are described. This chapter is a reasonably non-technical introduction to the model.

5.1 Overview of the conceptual model

The conceptual model conceptualises the 'world' as a series of *topologically connected discrete spaces*, each of which has a position, extent and orientation in 3D Euclidean space; thus, the model *combines* the *geometrical and topological* aspects of space. In addition, these spaces can be variously *grouped* to form the extents of various *real-world feature spaces* (parks, roads, courtyards, rooms, buildings). Spaces can also contain and be bounded by other features (e.g. walls and doors). The full geometry of the real-world is not the focus of the framework; rather it is the identification of and the topology between discrete spaces. This is very much in the GIS tradition and this thesis proposes that it is more appropriate to follow this approach for city-, region- or nation-wide 3D data than the approach of standard 3D modelling tools used by engineers and architects. In many countries, there is a great deal of good quality 2D mapping data. A system based on this conceptual model can exploit existing 2D data, instead of requiring an immediate and full detailed 3D survey.

This chapter draws heavily on the design principles presented in chapter 3. The framework can be organised into three parts:

- Geometry – a description of pure geometry in 3D space (Slingsby *et al.*, 2004a,b).
- Features – the attachment of real-world concepts to the geometrical description (Slingsby *et al.*, 2004a,b).

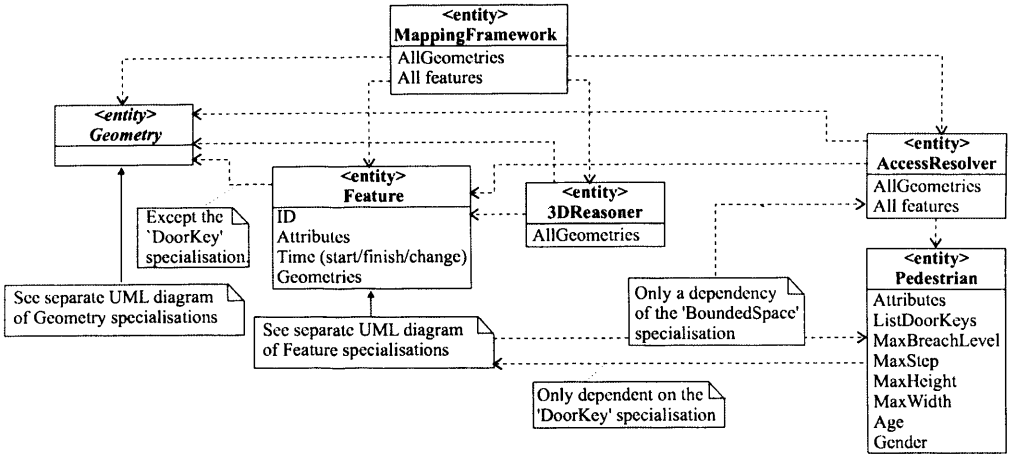


Figure 5.1: UML entity diagram showing the conceptual model. The six entities of 'MappingFramework', 'Geometry', 'Feature', '3DReasoner', 'AccessResolver' and 'Pedestrian' are shown. Dotted lines show dependencies. Specialisations of the abstract 'Geometry' entity and the 'Feature' entity are omitted from this diagram for reason of visual clarity; these are shown in figures 5.3 and 5.18, respectively. Note that the entities and hierarchies show do not need to correspond to the equivalents in any implementation of this conceptual model.

- Access – a description of how particular pedestrians with particular characteristics are able to move through space (Slingsby, 2005; Slingsby and Longley, 2006).

Figure 5.1 shows a summary overview of the conceptual model. Six concepts (entities) have been identified. These are:

- The 'MappingFramework' entity (section 5.2) is the entity through which the actors interact with the framework. It offers the functionality illustrated in the case-use diagram in figure 5.2. The 'MappingFramework' entity is dependent on the 'Geometry' and 'Feature' entity hierarchies and the '3DReasoner' and 'AccessResolver' entities.
- The 'Geometry' entity hierarchy (section 5.3) contains ten entity specialisations, shown in figure 5.3, and has no dependencies.
- The '3DReasoner' entity (section 5.4) is responsible for generating a 3D geometry from a set of 2D geometries and height constraints. It is additionally dependent on the 'Geometry' and 'Feature' entity hierarchies.
- The 'Feature' entity hierarchy (section 5.5) contains seven entity specialisations, shown in figure 5.18. All (except the 'DoorKey' specialisation) are dependent on the 'Geometry' entity hierarchy. The 'BoundedSpace' specialisation is dependent on the 'Pedestrian' and 'AccessResolver' entities.
- The 'Pedestrian' entity (section 5.6) is used to describe a type of pedestrian for the access delineation routines. It contains a set of pedestrian characteristics (including a set of door keys; thus is dependent on the 'DoorKey' specialisation of the 'Feature' entity).

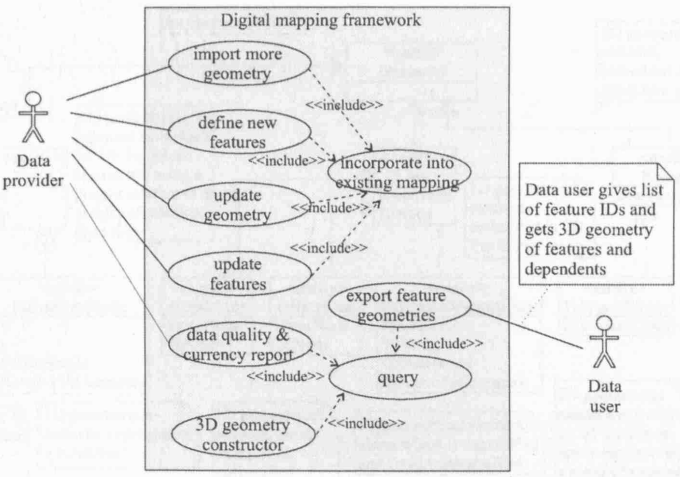


Figure 5.2: Use-case diagram, showing two 'actors' which interact with the framework in different ways.

- The 'AccessResolver' entity (section 5.7) resolves the spaces to which a specified pedestrian can gain access. It is dependent on the 'Geometry' and 'Feature' entity hierarchies and the 'Pedestrian' entity.

The concepts and the reasoning behind them will be described in the rest of this chapter.

Note that the entities and hierarchies, as depicted in figure 5.1, do not need to correspond to those of the entity equivalents in the implementation.

5.2 The 'MappingFramework' entity

The 'MappingFramework' entity is the entity through which all interaction with the framework is done, allowing different types of users to perform specific operations. Figure 5.2 is a UML use-case diagram, showing two different types of user ('actor'), the data provider and the data user. The *data provider* needs to be able to input, refine and update data, and also to assess the quality of the information. The *data user* needs to be able to retrieve features from the framework. The 'MappingFramework' entity allows such actors to interact with the database, but specific use-cases are not presented.

5.3 The 'Geometry' entity hierarchy

The abstract 'Geometry' entity and its nine specialisations are illustrated in figure 5.3. These entities represent the geometrical primitives which are used to model the pure geometry of the layers of the real world[1].

[1]Parts of this section are based on Slingsby *et al.* (2004a,b).

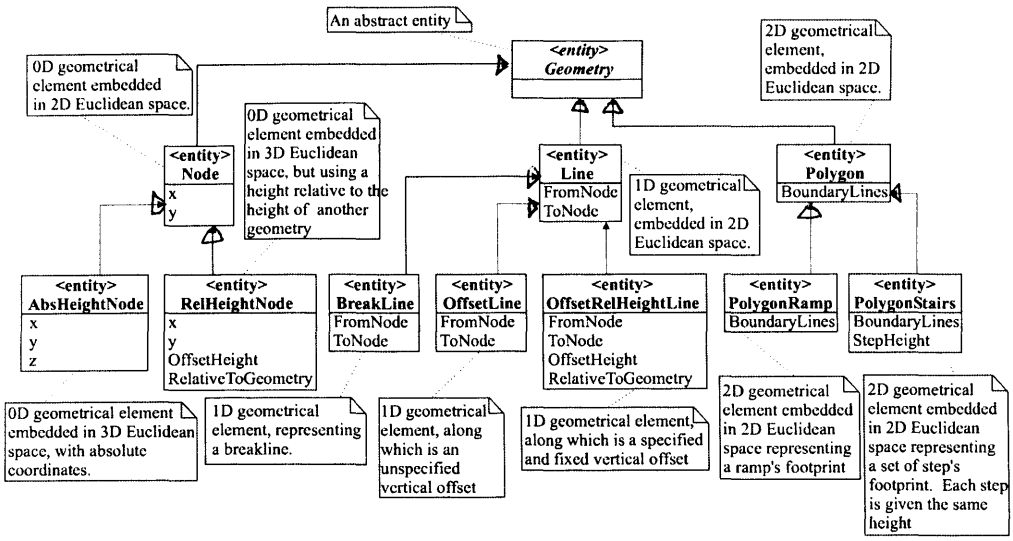


Figure 5.3: UML entity diagram for the 'Geometry' entity hierarchy, which fits into the UML diagram in figure 5.1. 0D, 1D and 2D geometrical primitives are defined. All of these entities describe their geometries in 2D Euclidean space, but two of the 0D primitives, 'AbsHeightNode' and 'RelHeightNode', describe their position in 3D Euclidean space (the latter is with respect to the height of another geometry). These geometries are topologically-connected in layers, within which it is intended that there are enough 0D primitives with heights for the 3D geometry to be reasoned by the '3DReasoner'.

Table 5.1: Geometrical primitive entities which affect the height and surface morphology of the surface in which they occur and where they occur.

Height constraints	**Surface morphology constraints**
'AbsHeightNode'	'BreakLine'
'RelHeightNode'	'OffsetLine'
	'PolygonRamp'
	'PolygonStairs'

The various geometry types within the 'Geometry' entity hierarchy are used to represent ground-surface topographical detail as a set of layers which are topologically connected into one multilayered model. Geometries within a single layer are non-overlapping tessellations, representing 'atomistic' (indivisible) primitives with no real-world meaning. Where a higher spatial resolution is required to describe the geometry of a feature, the geometrical primitives are broken down into smaller atomistic units, as illustrated in figure 5.20. Figure 5.1 shows that 'Geometry' is an abstract concept, and that the concrete geometry types are based on 0D, 1D and 2D geometrical elements. Each element describes its geometry in 2D Euclidean space, with the exception of 'AbsHeightNode' and 'RelHeightNode' which describe their geometries in 3D Euclidean space. The diagram also shows that they have no dependencies on any other entities. The combination of these geometrical elements in planes form layers, which in turn form the multilayered model.

As stated, all the 'Geometry' entities are described in 2D Euclidean space with the exception of the 'AbsHeightNode' and 'RelHeightNode' entities. The 'BreakLine', 'OffsetLine', 'PolygonRamp' and 'PolygonStairs' entities all provide surface mor-

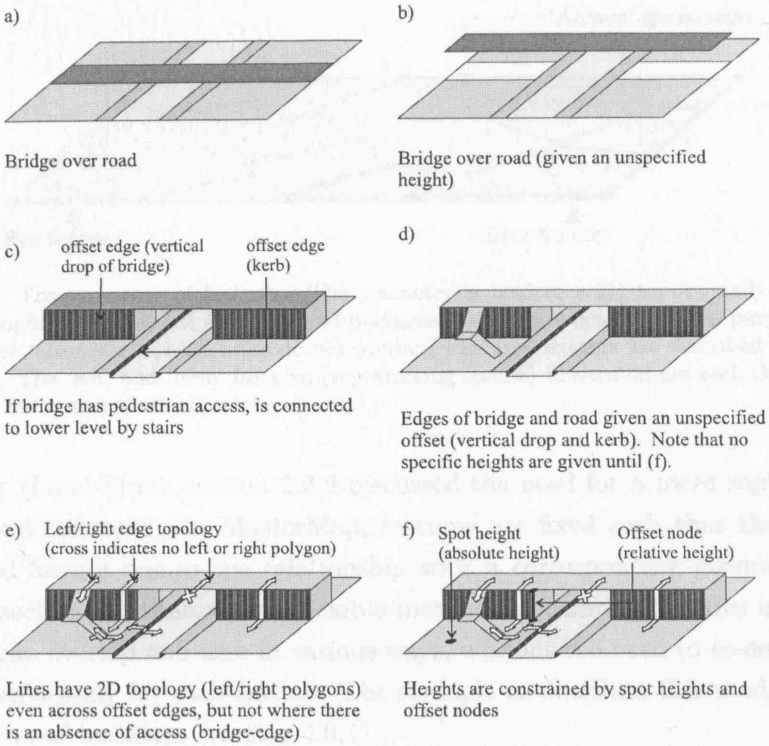


Figure 5.4: Illustration of the geometry of the proposed model.

phological information; as summarised in table 5.1. The '3DReasoner' will use the height and morphological information of these entities, along with well-defined rules, to generate a 3D geometry; as illustrated in figure 5.4.

Characteristics of the geometrical model are presented and explained in the sections below.

5.3.1 Separation of description of geometry and features

The framework models geometry and feature concepts separately.

5.3.1.1 Rationale

As discussed in section 2.6, GIS software tools are feature-based. The geometry of a feature is usually abstracted to a point (its position) or a polygon (its extent). The precise geometry of the feature is not represented because this is not the focus of GIS analyses. Traditional CAD systems tended to model pure geometry because they were designed to managed diagrams; later developments allowed these to be grouped into features. More recently developed parametric modellers are feature-based models, allowing their geometries to be precisely modelled (section 2.6.3 compared various aspects of CAD and GIS software packages).

OS MasterMap (being used as a case study) is a digital, large-scale, feature-based, vector data product. Although its feature-based approach is an improvement over its

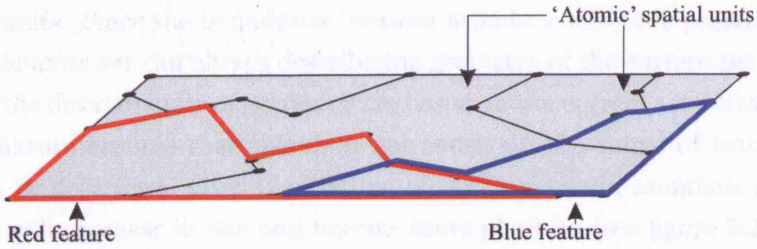


Figure 5.5: The geometry of features. The geometry is held in a 2D topologically-structured geometrical model of nodes, line segments and polygons (heights are ignored for the purpose of this diagram). The 'atomistic' spatial units do not overlap. Features' extents are described by a list of spatial units. The 'red' and 'blue' features (representing spaces) illustrated are each described by three spatial units, one of which overlaps.

predecessor (Land-Line), section 2.3.6 discussed the need for a more sophisticated feature-based database. In MasterMap, features are fixed such that they do not overlap and have a one-to-one relationship with a corresponding geometry. It is argued in section 3.1 that a more flexible method for defining features is required such they can overlap and nest in various ways, without the need to re-describe the geometry separately for each feature. The example to illustrate this need, concerns the definition of buildings (section 4.6.1).

The need to allow feature geometries to overlap or nest within each other in various ways, without re-describing their geometries, is the reason for the framework's separation of the description of geometry from the feature concepts themselves. Pure geometry with no real-world meaning has semantic meaning attached through real-world concepts ('features'). For this to work, however, the pure geometry needs to be discretised in some way (section 4.5). Usually, space is discretised based on the boundaries of the features of interest, but in this case, not all potential features of interest can be anticipated, support for geometrically overlapping features is necessary and the geometrical model should as independent from the feature model as possible.

5.3.1.2 Solution

The solution to this is to define the minimum set of non-overlapping geometrical primitives which is required to provide the geometry of all the features which exist in the database. Thus, the geometrical extent of many features is described by more than one of these 'atomistic' geometrical primitives; atomistic in so far as features cannot be described by *parts* of one of these. Thus, the geometry is discretised using all the boundaries of all *existing* features in the database, as illustrated in figure 5.20.

This appears to be in conflict with the principle that the description of geometry should be independent from that of the features. However, these atomistic units are *adaptive*. When a new feature extent needs to be added, if its geometry divides an atomistic unit, then the atomistic unit is permanently split into two smaller

atomistic units. Since the boundaries between atomistic units are preserved, the set of atomistic units are can always describe the geometry of the current set of features. Although the discrete units of geometry are based on the current set of features, their adaptive nature ensures that this does not constrain the range of feature extents which can be described. Over the lifetime of the framework, atomistic geometrical primitives will decrease in size and become more plentiful (see figure 5.3.5).

5.3.2 Basic 2D geometry and topology

The layers in the model are essentially modelled in 2D, with height and surface morphology information added. In section 3.5, '3D data' were defined as any data which have a description of a location in 3D space and perhaps a 3D extent such that their 3D positions and geometry in space is described. For working with (editing, validating and analysing) data, a fully resolved 3D model which can precisely model 3D positions and 3D topological relationships is a distinct advantage. For merely describing 3D data, less formal means can be used in which not all 3D relationships and descriptions are fully resolved. Although some of these can only describe 3D data which follows a predefined pattern (template), within this pattern, the 3D data can be described just as effectively and more concisely than more general-purpose 3D models. For example, if the 3D data being described have planes parallel to one of the 3 axes, a parameterised description which assumes this can describe the data just as well, but more concisely than a model which allows planes to be angled arbitrarily.

5.3.2.1 Rationale

2.5D models are routinely used for describing 3D data in a GIS context. 2.5D models allow a precise and detailed 2D descriptions of data with parameterised descriptions of information pertaining to the z axis. Such models are widely used because of their simplicity, the high quality 2D data already in existence, the proven adequacy of its use in many contexts, and the software limitations of handling 3D data.

This framework uses the 2.5D approach for the reasons above. The section concentrates on the 2D aspect of this, see the subsequent section for the 3D aspect.

The relevant design principles set out sections 3.4 and 3.5 are:

- the framework should provide a compact *description* of the real world
- initial data will come from existing 2D vector mapping data and spot heights (section 3.5)

5.3.2.2 Solution

As described, geometry is modelled as a set of 0D, 1D and 2D geometrical primitives in 2D space (with extensions to deal with height), with no real-world meaning, and

are organised into layers. Each spatial unit of the geometrical model is a 2D polygon, which implicitly represents the polyhedron of a 3D space, of which it forms the base. The geometrical entities are points, lines and polygons which are all topologically connected (if adjacent, and in the same layer or between layers) and do not overlap (if in the same layer).

5.3.3 Multilayering

In 2D vector topographic mapping, all topographic detail is represented on a planar layer. This can be draped onto a terrain elevation model. Space is implicitly modelled as that above the surface.

In this model, there are multiple layers, each of which is effectively draped onto its own terrain elevation model. The layers (composed of non-overlapping geometrical primitives) implicitly represent the space above. If there is a layer immediately above, this will implicitly represent the upper boundary of the space. Where this is the case, the 'thickness' of the layer is important; however, for reasons of simplicity, this is not discussed.

5.3.3.1 Reason

In a multilayered urban environment, overpasses, underpasses and different storeys of buildings overlap in two dimensions. When depicted on traditional maps, the uppermost layers obscure parts of the lower layers; this is clearly a problem with maps of multilayered environments. Maps (floorplans) of buildings and indoor shopping centres, usually break buildings up into storeys or parts of storeys. CAD software used for building plans does the same. However, in this framework, space should be treated in a seamless continuum and be handled independently of real-world meaning (e.g. independently of 'buildings' or 'storeys').

Since the layers are part of the geometrical model, they should be independent of the real-world meaning; layers should be defined using geometrical constraints only. The effect of this assertion is that for buildings, layers will not necessarily correspond to storeys (though they may happen to do so in many cases). Storeys do not have a consistent definition or numbering scheme (for example, split-level storeys may, in some cases, be treated as one storey and, in other cases, be treated as more than one storey) which varies between buildings. The geometrical model should store these independently of 'storeys', such that they can later be grouped into storeys (in the feature model) according to different criteria and have various numbering schemes attached.

As space should be treated seamlessly, all layers must be topologically joined.

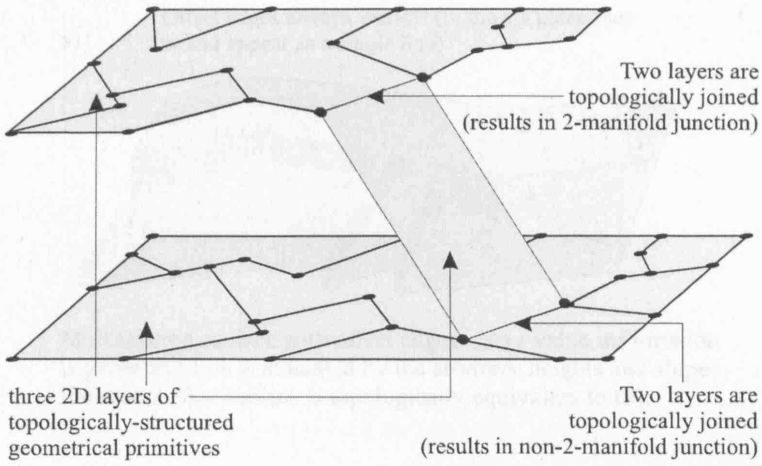


Figure 5.6: The joins between layers. Three areas are modelled in a 2D topologically-structured model. These areas have been topologically linked resulting in a 2-manifold join and a non-2-manifold join. The topological join allows the seamless traversal of the geometries.

5.3.3.2 Solution

To allow support for multiple layers, the framework allows separate layers to be topologically joined to each other. Some of these joins will be non-2-manifolds (this is the case for the base of the ramp illustrated in figure 5.6), so the geometrical model needs to support this situation. The resulting multilayered surface will be traversable, as if there was no seam. The layers themselves are defined and some using geometrical criteria; simply that each is a 2-manifold.

5.3.4 Height and surface morphology

No height information has been discussed so far, so the continuous multilayered surface cannot be oriented and placed in a 3D coordinate space. Figure 5.7 attempts to illustrate this by showing a continuous multilayered surface, arbitrarily placed in 3D space on the left and where height information has been added on the right.

This section describes how a surface (effectively a terrain model) can be built for each layer. Relevant design principles from chapter 3 are:

- the framework should cope with incomplete height data so that a model can be produced without the need for an exhaustive survey (section 3.4);
- the framework should be a data repository which holds and structures all available relevant data as they are collected, such that it is possible to build in an incremental fashion as data become available (section 3.4);
- any height information which affects access should be incorporated (section 3.3);
- relative heights are important (section 3.5), and this potentially affects access.

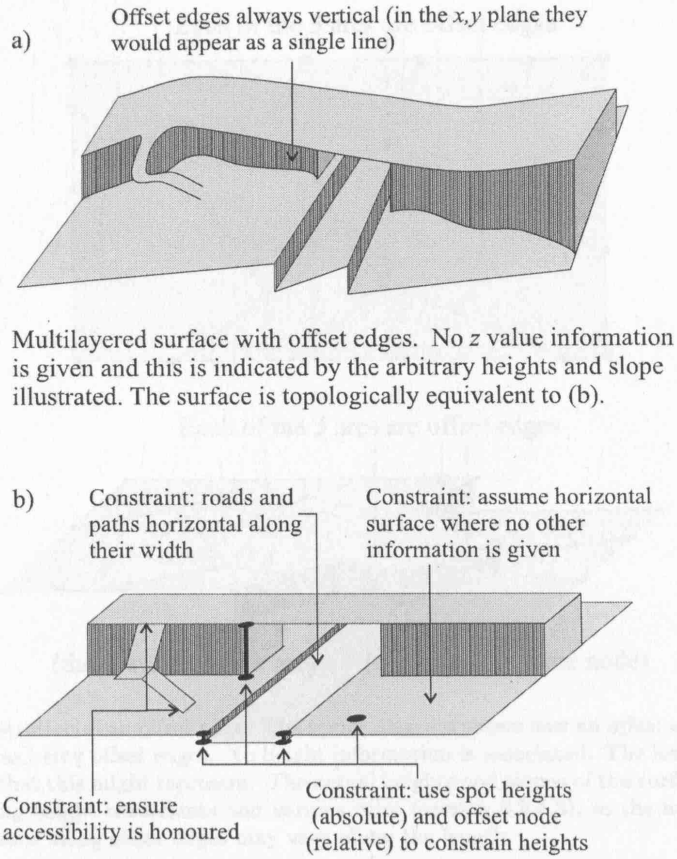


Figure 5.7: Illustration indicating how height and other constraints can be used to construct a 3D geometry.

Height and surface morphology constraints are embedded within the layers, from which a 3D geometry can eventually be generated (by the '3DReasoner' entity). As was summarised in table 5.1, the following height and surface morphology entities are defined:

- Breaklines (the 'BreakLine' entity) are used to indicate breaks of slope;
- Offset edges indicate vertical discontinuities in the morphology of the surface (also act as breaklines); the offset along the offset line may be variable;
- Absolute heights (the 'AbsHeightNode' entity) and relative heights (the 'RelHeightNode' entity; relative to other geometries) are scattered amongst the other geometries in the layers. These effectively 'pin' down points within the layers at specific heights;
- Polygons marked as ramps (the 'PolygonRamp' entity) and stairs (the 'PolygonStairs' entity).

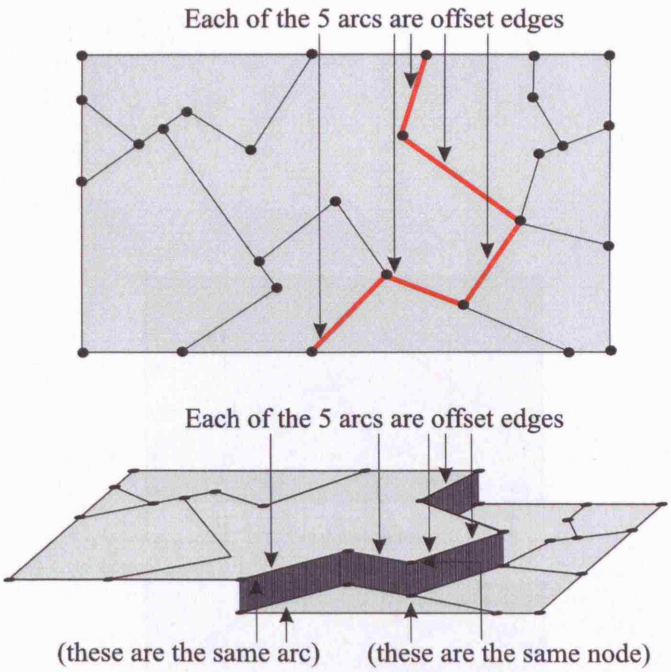


Figure 5.8: Illustration of an offset edge. The upper diagram shows how an *offset edge* is modelled; lines are marked as being offset edges. No height information is associated. The lower image shows the 3D situation that this might represent. The actual heights and slopes of the surfaces will depend on the surrounding height constraints and various rules (section 5.3.4.3), so the height differences at different positions along offset edges may vary along the length.

5.3.4.1 Breaklines (the 'BreakLine' entity)

Breaklines are lines which identify a break in slope in the surface.

5.3.4.2 Offset edges (the 'OffsetLine' entity)

An *offset edge* is a line along which the surfaces on either side meet at different heights. It thus represents a vertical offset in a surface, as illustrated in figure 5.8. The amount of offset is not specified – the resulting offset is a function of the heights described in the next section. As well at the vertical offset property, offset edges also function as breaklines because they represent a break in the slope between the surfaces on either side. Examples of real-world geometries modelled by offset edges are steps and kerbs. Figure 5.7 shows offset edges depicted by the vertical striped patterning corresponding to kerbs and the edge of the bridge. This figure also shows that the heights can be defined by 'pinning down' parts of the surface (and using certain rules); these are described in the next section.

5.3.4.3 Heights (the 'AbsHeightNode' and 'RelHeightNode' entities)

The 'AbsHeightNode' and 'RelHeightNode' entities are points (vertices) with height information attached. They are the only entities which assign heights. 'AbsHeightNode' specifies an absolute height and 'RelHeightNode' specifies a relative

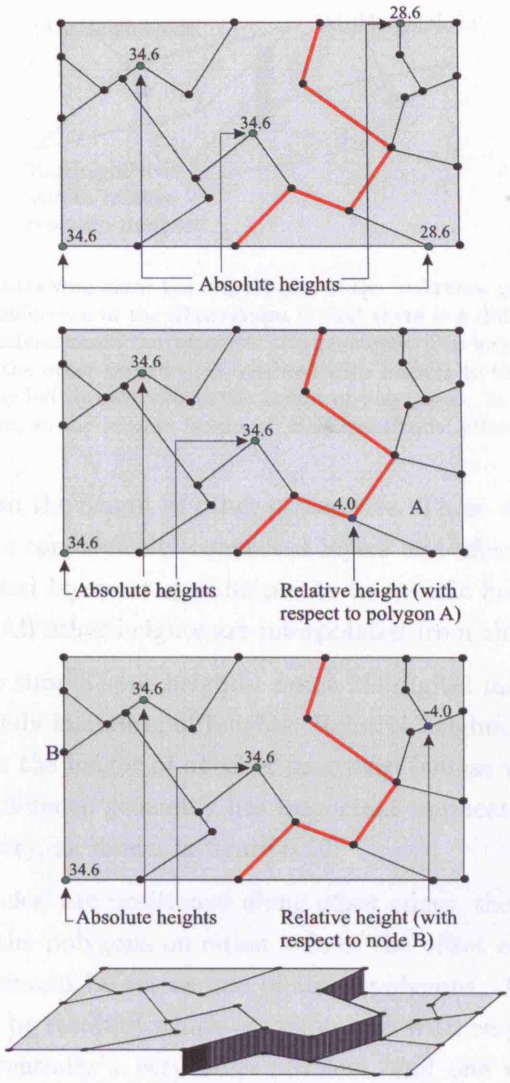


Figure 5.9: Illustration of how height constraints affect the surface geometry. The offset lines effectively separate the surfaces on both sides of the offset edge so that they behave independently. The three upper diagrams show how different combinations of height constraints can achieve the same 3D geometry (note than in this example, the heights are under-resolved – see section 5.4.1 and figure 5.12).

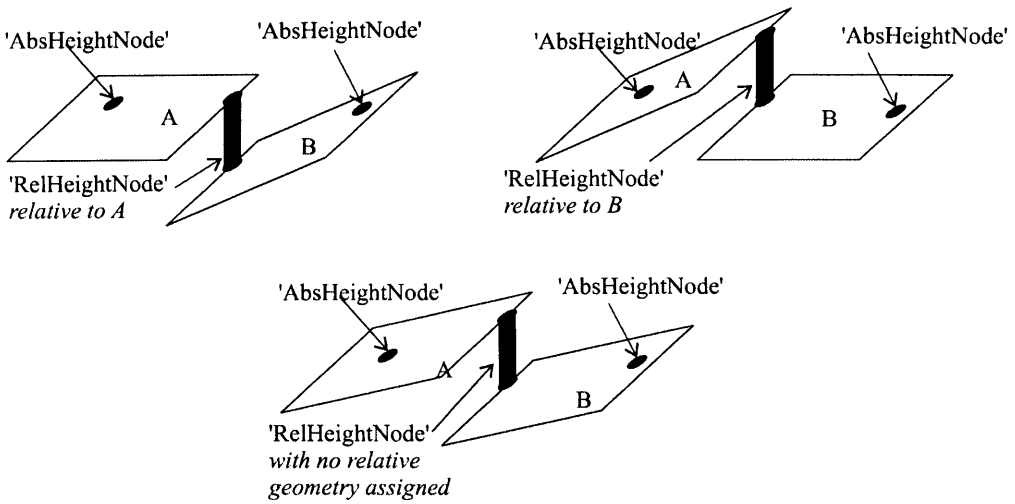


Figure 5.10: These illustrations show the importance of the 'reference geometry' of 'RelHeightNode' entities. The only difference in the illustrations is that there is a different *reference geometry*. This results in marked differences in the 'reasoned' 3D geometry. The height of the reference geometry remains fixed, and the other geometry is heighted with respect to the height of this. Thus in the illustration at the top left, it only affects the height of polygon B. In the lower illustration, no relative geometry is given, so the relative height of 'RelHeightNode' affects both polygons equally.

height with respect to the height of other geometries. These are scattered amongst the geometries of the topologically-connected layers and effectively 'pin down' the topologically-connected layers at specific points to specific heights.

All other heights are interpolated from these.

Absolute heights are simple spot heights. Some 2D digital mapping products (e.g. OS MasterMap) already include spot heights. Relative heights are heights which are given with respect to the height of another geometry (whose value may be positive or negative). This reference geometry has important implications for the resulting 'reasoned' 3D geometry, as shown in figure 5.10.

Where 'RelHeightNodes' are positioned along offset edges, they specify the vertical separation between the polygons on either side of the offset edge; in this case, the reference geometry should be set as one of these polygons. It is entirely possible that heights need to be resolved which are expressed with respect to other relative heights. This is potentially a very large problem, and one which is discussed in section 6.11.3.

Figure 5.9 shows how different combinations of absolute and relative heights can be used to achieve a 3D geometry. The heights in the example produce a horizontal surface; but undulating surfaces are just as easily described by the more dense use of nodes with heights attached.

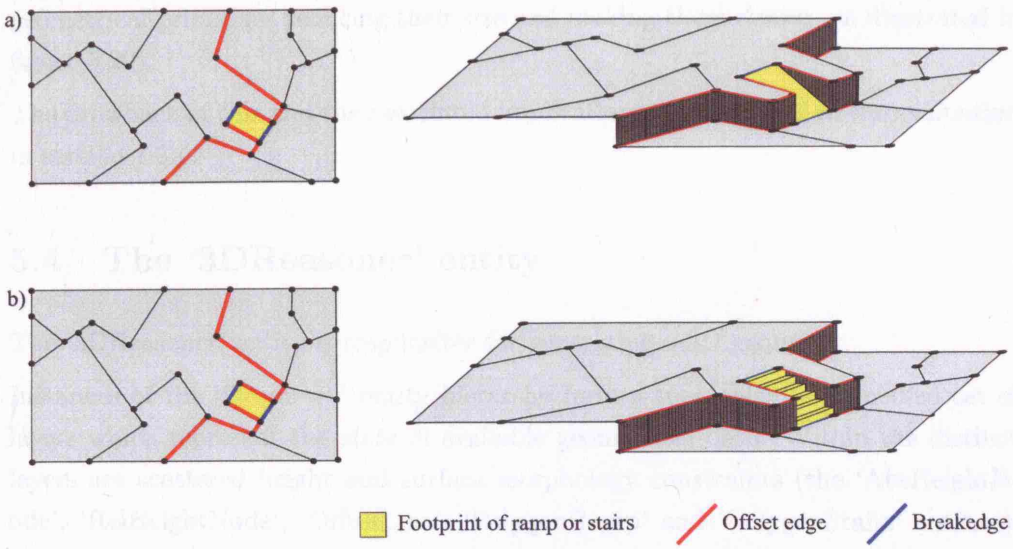


Figure 5.11: Illustration of 'PolygonRamp' and 'PolygonStairs' within a layer. On the left is the plan view, showing which lines around a 'PolygonRamp' and 'PolygonStairs' are 'BreakLines' and 'OffsetLines'. On the right, an example 3D view is shown.

5.3.4.4 Ramps and stairs (the 'PolygonRamp' and 'PolygonStairs' entities)

Polygons which are the footprints of ramps and stairs are specialisations of the Polygon entity. This is the case, despite the fact that they are part of the 'Geometry' hierarchy and as such should be defined completely independently of real-world concepts. Ramps and stairs are arguably real-world concepts; however, here, they are viewed as characteristics of the geometry of the surface morphology (as illustrated by their presence in table 5.1).

Instances of 'PolygonRamp' and 'PolygonStairs' entities always have breaklines at their tops and bases. Those which occur inline (within a layer), must have their sides bounded by offset edges (along which the height offset will vary), as illustrated in figure 5.11.

Each 'PolygonStair' instance has a step height attribute. The number of steps is a function of the difference in height between the top and bottom of the staircase and the height of each step.

5.3.5 Time and geometry

As discussed in section 3.6, time is an important consideration in a dynamic world. In this model, it is assumed that the pure geometry of the world is fixed, and that all changes to the geometry represent a *refinement* (rather than change) of the geometrical model by increasing its precision (the addition of more height constraints and surface morphology information) and its (2D) spatial resolution (by splitting

geometrical primitives reducing their size and making them denser, as illustrated in figure 5.20).

The drawback of this and the associated implications are discussed in the evaluation in section 7.3.4.

5.4 The '3DReasoner' entity

The '3DReasoner' entity is responsible for generating a 3D geometry.

Instances of the 'Geometry' entity hierarchy form a topologically connected set of layers which represent the *state* of available geometrical data. Within the distinct layers are scattered height and surface morphology constraints (the 'AbsHeightNode', 'RelHeightNode', 'OffsetLine', 'PolygonRamp' and 'PolygonStairs' entities). The '3DReasoner' entity generates a 3D geometry which conforms to these height and surface morphology constraints.

5.4.1 Interpolation from instances of the 'AbsHeightNodes' and 'RelHeightNodes' entities

This section concerns the role which instances of 'AbsHeightNodes' and 'RelHeightNodes' play in the generation of surface geometry. When a surface geometry is requested, it is *interpolated* from the height constraints in the 'Geometry' entity hierarchy (a mixture of absolute and relative heights modelled by the 'AbsHeightNode' and 'RelHeightNode' entities). Where there are enough, an *interpolation* technique is used to interpolate the surface geometry. Where they are not enough heights for interpolation, some *assumptions* are applied.

The situation as illustrated in figure 5.12 has very little of its area which can be properly interpolated, because only a small triangular area is completely surrounded with values to interpolate from. The height at any position within this triangular area can be interpolated. Outside this triangular area, heights must be allocated using an alternative method. Two approaches to this are:

- to extrapolate
- to assume horizontality

Extrapolation is an obvious solution. However, because the urban environment is dominated by horizontal planes (floors within buildings, car parks, paths, roads, pavements and are usually horizontal, at least perpendicular to the direction of travel), it might be more appropriate to assume horizontality. It should be noted that the ideal case is that any position on the surface can be interpolated; applying these assumptions is only done where the data are under-resolved for interpolation. The way in which these rules are applied, is illustrated in figure 5.13.

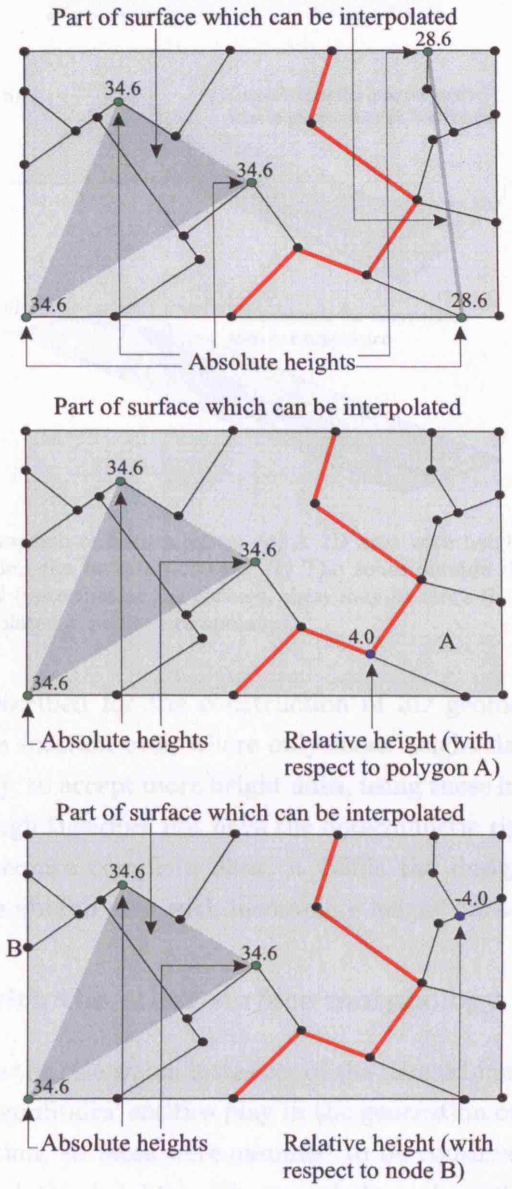


Figure 5.12: Height interpolation within patches. The dark grey areas show which parts of the surface can be interpolated.

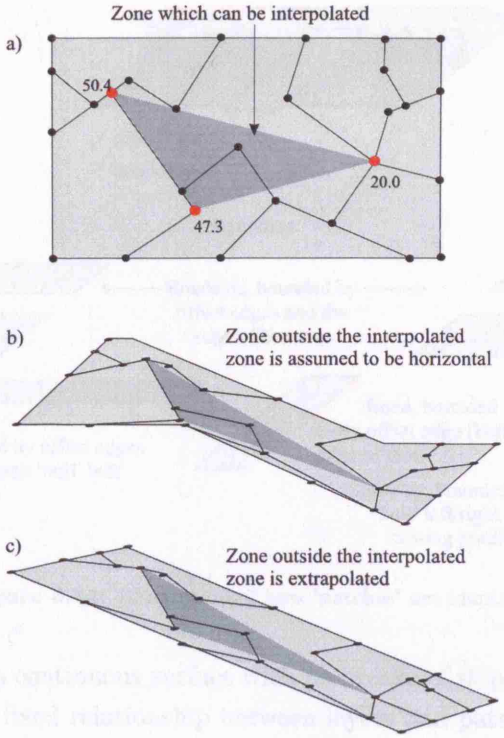


Figure 5.13: The assumption of horizontality. *(a)* A 2D area with height constraints. The dark grey shows the zone which can be interpolated. *(b)* The zones outside the interpolated zone are assumed to be horizontal (note that at the corners, there may be more than one solution). *(c)* The zones outside the interpolated zone are extrapolated.

These techniques described for the construction of 3D geometry are designed to produce a reasonable solution even where only scant height data are available, but are able, subsequently, to accept more height data, using these improve the generated 3D geometry. Although this does not have the deterministic rigor of more orthodox 3D models, which require complete data, it fulfils the design principle that the framework should be able to deal with incomplete height data.

5.4.2 Dealing with the other surface morphology information

This section concerns the role which instances of the 'BreakLine', 'OffsetLine', 'PolygonRamp' and 'PolygonStairs' entities play in the generation of surface morphology (in the previous section, surfaces were assumed to be continuous and interpolated using absolute and relative heights). The morphological constraints listed in table 5.1 are used to break the surface up into a *patchwork of continuous surfaces* (with the exception of 'PolygonStairs', whose stepped surface is generated separately), each of which is called a *'patch'*, *between which* are breaks of slope ('Breaklines') and vertical discontinuities ('OffsetLines'). Each set of contiguous 'PolygonRamp' and 'PolygonStairs' instances comprises its own patch; each dedicated 'PolygonRamp' and 'PolygonStairs' patch directly generates its own surface geometry.

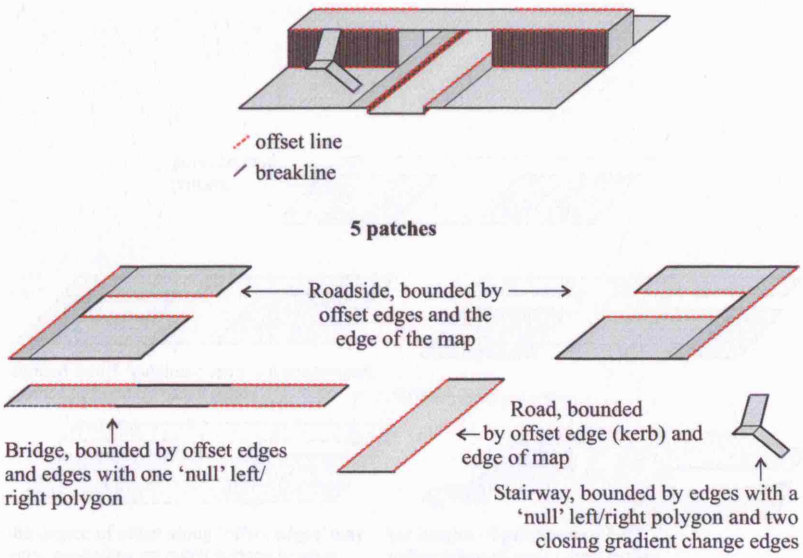


Figure 5.14: Illustration of how 'patches' are identified.

A 'patch' is part of a continuous surface with no breaks of slope or vertical discontinuities. There is no fixed relationship between layers and patches, but a patch will not contain more than one layer (because of potential self-intersection) and is likely to be part of a layer (because of offset edges and breaklines). Patches are can be delineated automatically, by finding all the sets of connected geometrical primitives which are bounded by instances of 'BreakLine' and 'OffsetLine' entities, or which are instances of 'PolygonRamp' or 'PolygonStairs' entities. Thus, patches are intended to only be used internally by the framework for generating 3D geometry.

The concept of patches is not exposed to the user. They are intended to be generated automatically, behind-the-scenes, from the seamless geometry in the database, and only be used, internally, by '3DReasoner'. Figure 5.14 illustrates how a scenario containing a bridge, is automatically broken up into patches; most of the patch surfaces are shown are horizontal, for simplicity of illustration.

The surface geometry of a patch only interacts (in a height sense) with the surface geometries of adjacent patches where there is an instance of 'RelHeightNode' between these patches (which specifies a height separation between them). An 'OffsetLine' (which has no specified height) only causes a height interaction if there is a 'RelHeightNode' along it. Figure 5.15 shows that this restricted set of interaction points between patches can result in some quite complex geometrical relationships between different parts of the patch, even though the way in which heights are specified is relatively simple. This is important in cases where different levels weave in and out of each other. An example is the railway line in figure 2.3 (the dark grey polygon stretching diagonally from the bottom left of the figure), which is cut by a road at the bottom left of the figure (indicating that it bridges a road) and is cut by five roads towards the right of the figure. These roads are on a separate patch to the railway and the railway patch weaves under five roads and over one.

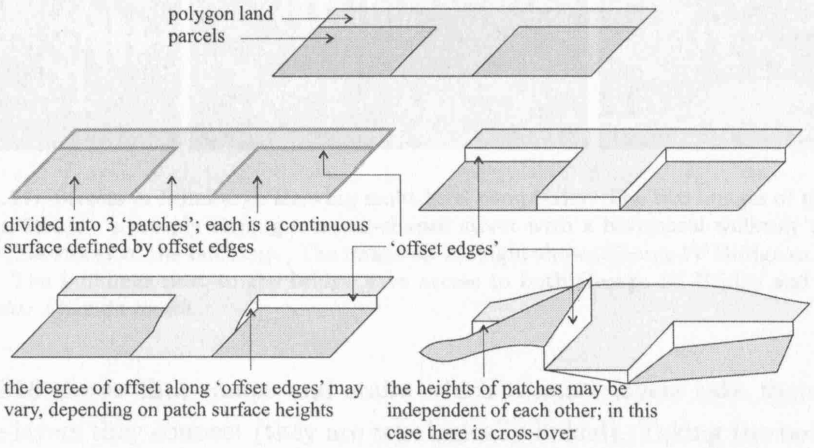


Figure 5.15: These illustrations show that the height relationships between patches may be quite complex. Notice that offset lines may have an offset which varies along the length or even crosses from a negative offset to a positive offset. The 3D geometry of these examples is dependent on the position of absolute and relative heights (not shown).

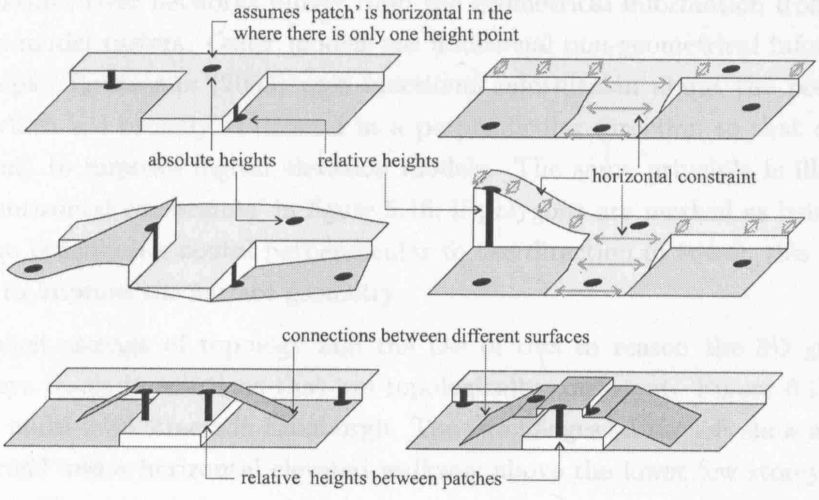


Figure 5.16: Example of how the surface heights of patches interact when 'pinned down' by height constraints (section 5.3.4.3 and where rules are applied (section 5.4.1).

Figure 5.17: Streets in Edinburgh showing multi-level complexity. The two images of the left are of Victoria Street – a steeply sloping crescent-shaped street with a horizontal walkway across the ground or first storey of the buildings. The image on the right shows George IV Bridge as seen from Cowgate. The buildings next to the bridge have access to both George IV Bridge and Cowgate. *Photographs: Mike de Smith.*

Figure 5.16 shows that ramps and stairs which connect layers take their heights from the layers they connect (they are topologically linked). Taking the bottom-left example in the figure, a change in the relative heights would result in a change in the gradient of the ramps leading up to the bridge. This topological connectivity is very important, because where poor height data exist, the 3D 'reasoned' solution will be topologically consistent.

If the *geometrical model* has access to a *feature model* (with real-world meaning), there is scope for using the properties of the *functions* of features to further enhance the likely accuracy of the 3D geometry generated. Similar techniques are employed in terrain models which use information about the terrain other than pure geometrical information in order to interpolate a better surface. Some algorithms attempt to extract this information directly from the geometry; for example, Fowler and Little (1979) extract river networks purely from the geometrical information from digital elevation model rasters. Other models use additional non-geometrical information; for example, Rousseaux (2003) uses functional information about the position of roads (which are broadly horizontal in a perpendicular direction to that of traffic movement) to improve digital elevation models. The same principle is illustrated as the 'horizontal constraints' in figure 5.16; if polygons are marked as being roads which are broadly horizontal perpendicular to the direction of travel, this fact can be used to improve the surface geometry.

The explicit storage of topology and the use of this to reason the 3D geometry will always result in solutions that are topologically consistent. Figure 5.17 shows complex multi-level streets in Edinburgh. The two images on the left show a steeply sloping road and a horizontal elevated walkway above the lower few storeys of the buildings. The top of the road and the walkway join at George IV Bridge (not shown) at the same level. The image on the right shows a road bridge adjacent to buildings. There is access to some of the buildings from both the base of the bridge and the top of the bridge. The storage of these topological relationships ensures a topological consistency in 3D model output, even if there is incomplete height

data. Thus, the '3DReasoner' will use the captured topology, height constraints and surface morphology constraints to generate a reasonable 3D output which conforms to these.

The approach of interpolating z values from poor or incomplete height data and of incremental updating is in marked contrast to most 3D models and surveying projects, where every coordinate is 3D (holding a z value in addition to the x and y values). For modelling cities over such large areas, this approach may be a more sensible initial approach than full 3D. A fully-resolved 3D approach would require every coordinate to be allocated a height value, even if this has not actually been surveyed. Exhaustively surveying all would be an expensive and time-consuming process (at a sub-metre resolution, including underpasses and inside buildings).

Although the concept of the incremental addition of detail appears to be in contrast with the strict mapping specifications enforced by national mapping agencies, such mapping specifications are part of policy rather than a fundamental part of a framework. Such a policy could still be enforced; e.g. to ensure that the addition of new data conformed to one of a set of well-defined capture specifications.

5.5 The 'Feature' entity hierarchy

This section[2] describes the 'Feature' entity hierarchy which is illustrated in figure 5.18. Features (other than 'DoorKey' which is not geometrical) group sets of geometrical primitives, giving them 'real-world' meaning. In other words, a feature represents a conceptualisation of the real-world feature, whose extent is defined by the union of a (usually fixed) set of geometrical primitives, as illustrated in figure 5.19.

As shown in figure 5.18, there are six specialisations of the 'Feature' entity (plus 'Feature' itself). Unlike in the geometry entity hierarchy, the base entity of 'Feature' is not abstract; instead it represents the simplest feature type. The specialisations of 'Feature' possess additional feature-specific information. The six specialisations presented here are certainly not exhaustive. They have been chosen to illustrate the types of specialisations of feature which can be defined, but particularly in order to address the pedestrian access issues developed in this thesis. Clearly, different application domains will require different specialisations of 'Feature'.

5.5.1 Geometry of features

All the features in the 'Feature' entity hierarchy have a geometrical extent, with the exception of the 'DoorKey' entity.

[2]Parts of this section are based on Slingsby *et al.* (2004a,b)

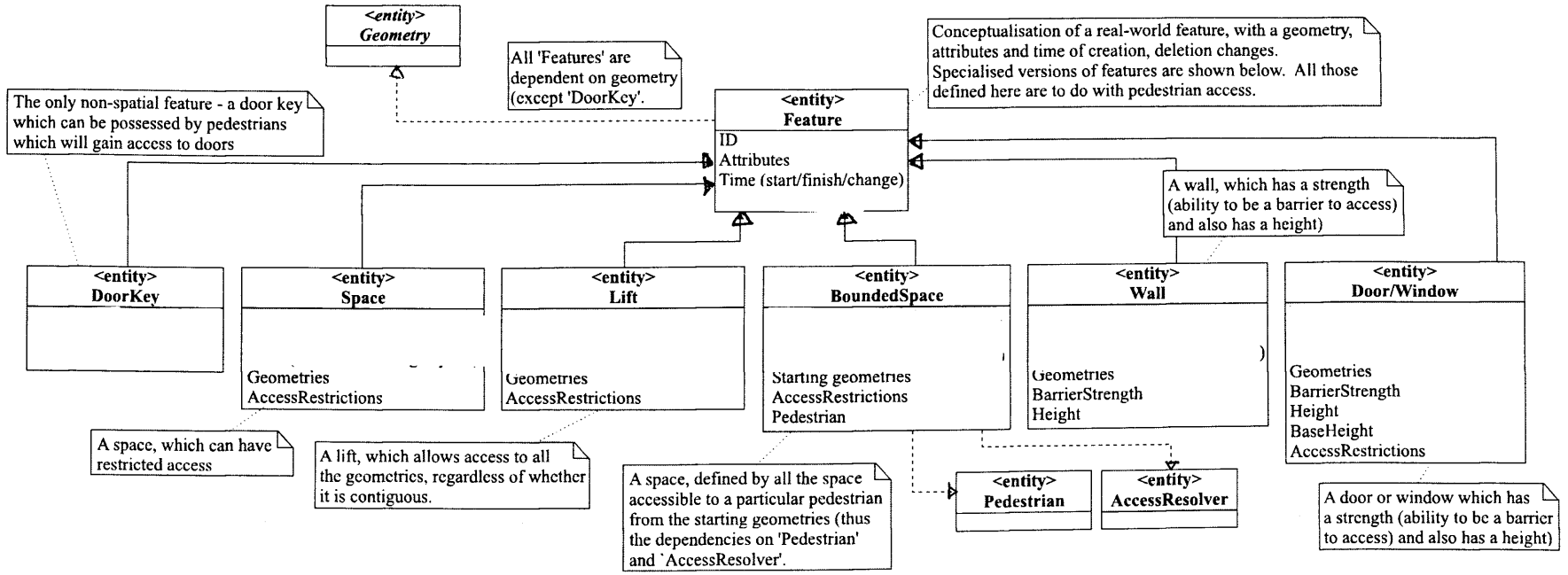


Figure 5.18: UML entity diagram for the 'Feature' entity hierarchy (showing direct dependents), which fits into the UML diagram in figure 5.1. This shows the seven types of feature defined for the implementation. Every feature (except 'DoorKey' which is not a geometrical feature) has a time-stamped geometrical extent, described with a set of geometrical primitives (hence the dependency on 'Geometry'); for most features, this is fixed, but for 'BoundedSpace', it is an access-based query, specific to a pedestrian (hence the dependency on 'Pedestrian' and 'AccessResolver'). Each feature type is self-contained and any number of feature types (specialisations of 'Feature') can be added; the six specialisations defined here have been done so because they are required for demonstrating the concepts of this thesis; particularly access.

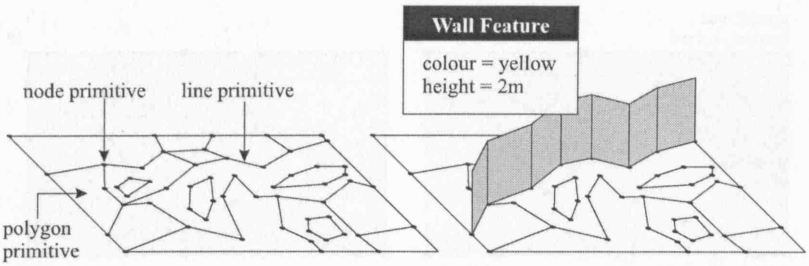


Figure 5.19: The geometry of an addressable object is described with a list of the comprising geometrical primitives. In this example, the geometry of a 'Wall' feature is described by a list of seven line primitives. The feature also has attributes.

5.5.1.1 Features with fixed geometrical extent

All the features, with the exception of 'DoorKey' (it is not geometrical) and 'BoundedSpace' (see following section) have their extents described with fixed sets of geometrical primitives which form the extent of the 2D footprint. The 3D geometry is parameterised by the feature; for example, 'Wall' has a height attribute.

Membership of geometrical primitives to a feature's geometrical extent is time-stamped. Although a distinction should be made between different types of time (e.g. database time and world time; section 4.8), for the purposes of simplicity, only one type of time is considered here. The time-stamping allows the geometrical primitives, which comprise the feature's extent, to be valid for certain ranges of time; whether the time corresponds to the change in the feature's real-world existence, when the change was noticed, when the change was recorded, or more than one of these, is dependent on the type of time used in the implementation. For example, if a building comprises ten polygons, the addition of a building extension may add a further two polygons to the feature's extent, whose timestamps of membership to the building's extent record a validity from the time the extension was built, noticed or recorded. This concept is illustrated and explained in figure 5.20.

5.5.1.2 Features whose extent is access-dependent

Some features' geometrical extents, instead being described by a fixed and prescribed set of timestamped geometrical primitives, have their extents described by other means. 'BoundedSpace' is an example of one of these, whose extent is described by an access-dependent criterion. The criterion is based on one or more starting points and a description of a pedestrian. The resolved geometrical extent of the feature is the space accessible to the described pedestrian from the starting points, as resolved by the 'AccessResolver' (section 5.7). Because of the nature of the access model, the geometrical extent of the feature may vary, depending on the time at which the geometrical extent is resolved.

The concept of a space feature whose geometrical extent can be described in terms of accessible space is inspired by the observation that there appears to be a relationship

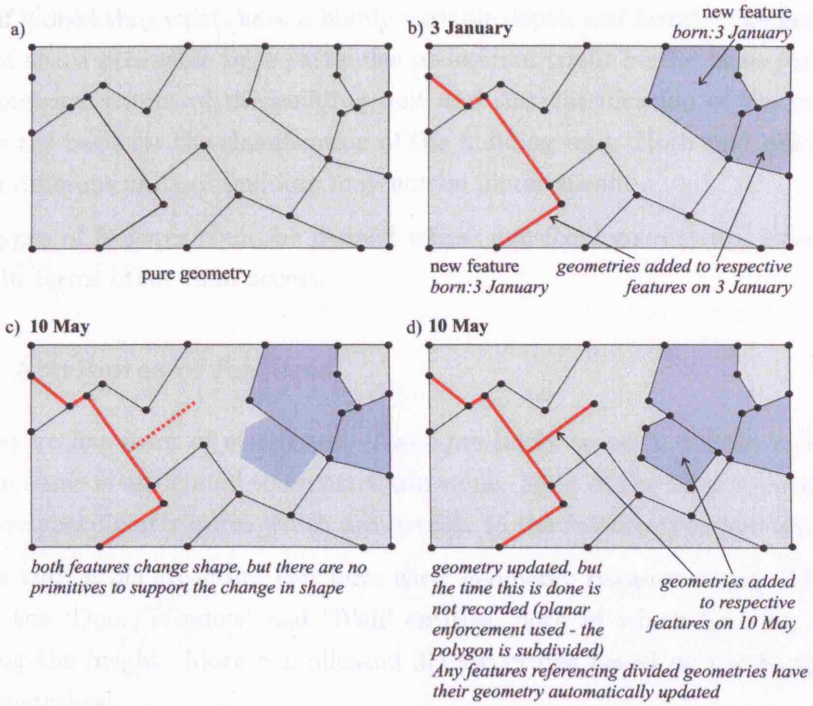


Figure 5.20: Time-stamped features. *(a)* Shows geometrical primitives which are not referenced by any features. *(b)* Two features are created (a linear feature and an areal feature). The time at which the feature was created and the time at which the geometries primitives were added is recorded. *(c)* The features have a change in their geometrical extents, but the geometrical primitives do not support the new geometry required. *(d)* This is resolved by splitting the existing primitives to add the required geometrical primitives. The time at which the geometries are split is *not* recorded (time is not recorded for the geometrical primitives and the splitting corresponds to a refinement – a local increase in the 2D spatial resolution), but the time at which these new primitives are added to the feature geometries *is* recorded. This allows the extent of the feature to be reconstructed before and after the change.

between an activity or function of a space and access to that space. Features with their extents described in this way are conceptualisations of spaces which fulfil a specific function or within which there is a specific activity, using access as a proxy for activity or function. Figures 4.17 and 4.18 show examples of the relationship of access and function in buildings. In office environments, it is likely that groups of offices can be distinguished by the patterns of access authorisation. Another example is a block of flats, where different parts have access to progressively smaller groups of people as the hierarchical depth increases from outside to inside; to illustrate this, consider a block of flats containing three groups of spaces (in terms of access):

- space communal to all residents of the block of flats;
- space communal only to residents of a flat;
- space communal only to inhabitants of a room in a flat.

One can identify a hierarchy of three levels, corresponding to different groups of people; access to communal parts of the block from the main door, access to this plus a resident's flat's communal space accessible from the flat's main door and access all these spaces plus a resident's room within the resident's flat. (Such hierarchies of

access, if indeed they exist, have a highly variable depth and form). The geometrical extent of space accessible by a particular pedestrian could be the basis for defining the geometrical extent of the building unit and the classification of the pedestrian could be the basis for the classification of the building unit. Note that relationships between different units of building may not be hierarchical.

Other types of features could be defined whose non-fixed geometrical extent is described in terms other than access.

5.5.2 Attributes of features

Each feature has a set of attributes. These are likely to be in a form in which an attribute name is associated to an attribute value. Most of the more specialised features have specific attributes which are specific to the feature-type and its purpose.

Features with a 3D structure can have their geometry parameterised. This is the case for the 'Door/Window' and 'Wall' entities, both of which have an attribute indicating the height. More complicated 3D structures based on the footprint can be parameterised.

Feature attributes need to be timestamped in the same way that geometrical primitives' membership of the geometrical extent of features is timestamped.

5.5.3 Time and features

As stated, time is of no concern to the 'Geometry' entity hierarchy; it is assumed that any change is a refinement or an increase in spatial resolution (section 5.3.5). Geometry is seen as an invariant description of pure geometry which is not affected by real-world change. However, since features correspond to real-world conceptualisations, time is of utmost importance, and is represented as the 'time' attribute in the entity diagram shown in figure 5.18. However, this attribute obscures the multiple aspects of the feature for which time must be recorded. These are:

- the time over which the feature is in existence; creation, destruction and temporary suspensions of existence;
- changes in the geometrical extent of the feature;
- changes in the attributes of the feature.

The latter two of these have already been described and they use snapshot time (a description of an instance in time; section 4.8.1) for timestamping.

The existence of a feature in time can be described by both snapshot time and recurring time. Recurring time (cyclic time) recurs through time in a cycles of various lengths; i.e. daily, weekly, monthly or yearly cycles. This is appropriate for describing features' changes where these are cyclic. For example, bollards blocking access to traffic may only be in use at certain times in the day or week on a recurring

basis. Features do not necessarily need to have a physical presence; the congestion charging zone in Central London has a well-defined spatial extent, but it only 'exists' within a daily cyclic and a weekly cycle.

5.5.4 The 'Feature' entity

The 'Feature' entity is the base class of the 'Feature' hierarchy and is thus the simplest feature type. It has a basic existence, a prescribed geometry and a set of attributes. Changes to geometry and attributes are time-stamped to allow their reconstruction for any time in the past.

5.5.5 The 'DoorKey' entity

The 'DoorKey' is the only non-geometrical entity; instead of geometries, it can hold the 'Door/Window' features for which it has access. It represents a door key, access card, ID card or any other authorisation which allows access through a door. Instances of the 'Pedestrian' entity can possess 'DoorKeys' and this allows them to gain access through the 'Door/Windows' which require one of the possessed 'DoorKeys'.

5.5.6 The 'Space' and 'BoundedSpace' entities

The 'Space' and 'BoundedSpace' entities represent spaces and thus tend to be composed of geometrical primitives which are polygons. The 'Space' entity has a prescribed set of geometrical primitives, whereas the geometrical extent of 'BoundedSpace' is access-dependent using the 'Pedestrian' and 'Starting geometries' attributes (seen in the UML diagram in figure 5.18) as described in section 5.5.1.2.

Both have an 'AccessRestrictions' attribute. This allows them to affect the passage of pedestrians, in the same way that 'Door/Window' entities do. The 'AccessRestrictions' attribute is a list of the conditions under which a pedestrian can have access; this is explained in section 5.7.1.4.

5.5.7 The 'Lift', 'Wall' and 'Door/Window' entities

The 'Lift', 'Wall' and 'Door/Window' entities are important in terms of pedestrian access. They have parameters specific to their purpose. 'Lift' and 'Door/Window' entities have 'AccessRestrictions' attributes (as do 'Space' and 'BoundedSpace' entities); this attribute is not appropriate for 'Wall' entities, because they are barriers to access, not facilitators ('Door/Window' and 'Lift' entities are both barriers and facilitators of access).

'Lifts' are space features (i.e. composed of polygons), in which comprising polygons (in a vertical stack) are topologically connected. They are used to allow access

between polygons in different layers which are not directly topologically connected through geometrical primitives in layers. 'Lifts' are used by 'AccessResolver'.

'Walls' are barriers to access. Their footprints may be represented by line or polygon primitives. One 'Wall' is assumed to have a constant height along its length, described by the 'Height' attribute. 'Walls' have a 'BarrierStrength' attribute, which represents the physical capability of a barrier to block access where it is not allowed. This is fully described in section 5.7.1.4.

The 'Door/Window' entity represents either a door or window. Like 'Wall' entities, their footprints are described with a set of line or polygon primitives. The same entity is used for both types of real-world feature, because, for the purposes of this thesis, their main function is to control access. Also, in some cases, windows are used for access (e.g. a windows which has been left open), and some windows function as doors (e.g. French windows).

The 'Door/Window' entity has the same attributes as 'Wall', with the additional of:

- a 'BaseHeight' attribute – the height between the ground surface and the base of the feature;
- an 'AccessRestrictions' attribute (as the 'Space', 'BoundedSpace' and 'Lift' entities).

The only differences between doors and windows in terms of this entity's attributes, is that windows are usually higher off the ground (the 'BaseHeight' attribute) and are less likely to have 'DoorKeys' which can open them.

5.6 The 'Pedestrian' entity

Parts of this section are based on Slingsby (2005); Slingsby and Longley (2006). The 'Pedestrian' entity is a description of the state of a pedestrian. It is used exclusively by the 'AccessResolver' entity (section 5.7) and its attributes are dependent on the needs of this. The attributes shown for this entity in figure 5.1 are listed below:

- Attributes – characteristics about the pedestrian;
- 'ListDoorKeys' – a list of all instances of the 'DoorKey' entity possessed by the pedestrian;
- 'MaxBreachLevel' – the extent to which a pedestrian might gain access to a feature when the 'AccessRestrictions' attributes do not allow this;
- 'MaxStep' – the maximum step height a pedestrian is able to negotiate;
- 'MinWidth' and 'MinHeight' – the minimum space through which the pedestrian can pass (this will also depend on the size of luggage being carried);
- 'Age';
- 'Gender'.

These are explained in the next section, so are not expanded here. Again, these attributes are by no means exhaustive; here, they are dependent on the detail of the pedestrian access model.

5.7 The pedestrian access model and the 'AccessResolver' entity

This section will describe the pedestrian access model. The *state* of pedestrian access is encapsulated by the 'Geometry' entity hierarchy (and the geometry reasoned by the '3DReasoner' entity), the 'Feature' entity hierarchy and the 'Pedestrian' entity. The *process* of pedestrians accessing space is encapsulated by the 'AccessResolver' entity. The 'AccessResolver' entity's job is to delineate all the space starting from one or multiple points which is *accessible to a pedestrian with specific characteristics at a specific time.* Although it is pedestrian access which is considered here, it is anticipated that the same concepts could, in principle, also be used for vehicular access.

Pedestrian accessibility is delineated in a piecemeal fashion using a specific instance of a 'Pedestrian' entity and a specific snapshot time. Starting at a specific geometry, the ability for the specific pedestrian to have access is assessed (see subsequent section). If access is successful, the geometry is added to the working set of accessible geometries. Each of these geometries has all its topologically-connected geometries identified (connected in the layer or through a 'Lift'). All of these which allow access are added to the working set. This continues until all the geometries in the working set have been assessed for access. The result is the union of the geometries which define an accessible space.

5.7.1 Four factors affecting pedestrian accessibility

The pedestrian access model is dependent on four factors (figure 5.21), the *states* of which are held as attributes distributed amongst the 'Geometry' entity hierarchy (and the geometry reasoned by the '3DReasoner' entity), the 'Feature' entity hierarchy and the 'Pedestrian' entity. These are interpreted by the 'AccessResolver' entity.

5.7.1.1 The pedestrian

The 'Pedestrian' entity encapsulates the relevant aspects of a pedestrian for the 'AccessResolver' entity. Through its attributes, it describes the characteristics of a pedestrian.

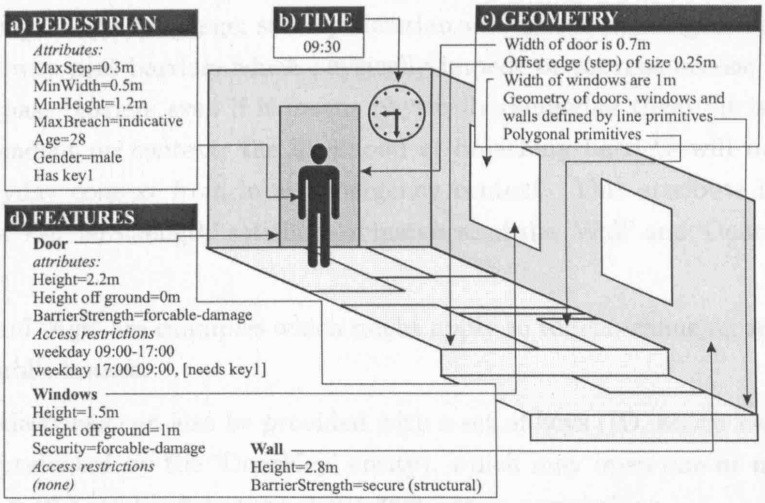


Figure 5.21: Access is dependent on four factors: a) attributes of the pedestrian; b) the time; c) the geometry and d) attributes and access restrictions on features. As the geometrical model is essentially 2D, the door and window heights are stored as feature attributes rather than being described by the geometrical model. Walls, doors and windows in this example are composed of line primitive segments.

'MaxStep' indicates the maximum step-height that the pedestrian can negotiate. For a typical pedestrian this might be set at 0.3m, a value in excess of 1m might be set for an agile and determined pedestrian, while for a wheelchair-bound individual, this might be set to zero. This value is used to assess whether the pedestrian can gain access through features and geometries. In terms of features, 'Door/Window' entities (depending on the 'BaseHeight' attribute value) and 'Wall' entities (depending on the 'Height' attribute value) affect potential access. In terms of geometries, the minimum height of an 'OffsetLine' bounding a polygon and the 'StepHeight' attribute of 'PolygonStair' is assessed. As stated, it is likely that the main difference between doors and windows (within the 'Door/Window' entity) is that windows tend to be higher off the ground surface. When reached, however, they can serve as access points; usually in the context of either unauthorised or emergency access. Hence burglars and the emergency services may be similar insofar as they have large 'MaxStep' values.

'MinWidth' and 'MinHeight' indicate the minimum width of space which the pedestrian can pass and the minimum headroom required. In particular, wheelchair users and pedestrians carrying bulky equipment will require a wider width of space for passage. This is used to assess the gap between barriers (e.g. corridor width), and the width and height of doors (and windows).

'MaxBreachLevel' indicates the maximum security level (section 5.7.1.4) of a wall, door or window that the pedestrian is able or willing to breach to gain access, if they are not allowed access by the access restrictions. Its aim is to encompass a range of barrier strengths, ranging from the types of queuing tapes found in Post Offices and unlocked 'staff only' doors (i.e. barriers which indicate, rather than physically prevent access) to secure doors of the type found in bank vaults. Some pedestrians

will normally obey all signage; some pedestrian will ignore such signs, but will not force their way past barriers which physically impede or prevent access; others are willing to pass barriers even if it means physically damaging them. It is also very much dependent on context; the likelihood of breaching barriers will be different in an everyday context from in an emergency context. This attribute is assessed against the 'BarrierStrength' attribute of instances of the 'Wall' and 'Door/Window' entities.

'Gender' and 'Age' are examples which might apply to toilets, changing rooms, play areas or public houses.

Finally, pedestrians can also be provided with a set of keys (ID, access card or door key; all represented by the 'DoorKey' entity), which may open one or more doors and which may be subject to time-dependent access permissions.

5.7.1.2 Time

Access is delineated according to a pedestrian with specific characteristics and a specific (snapshot) time. The time is used to assess access in three ways:

- To ascertain whether any feature being queried is actually in existence. For a feature to be in existence the time must be after the 'creation' timestamp, before the 'destruction' timestamp (if one exists) and within the part of the time cycle in which is it is existence (section 5.5.3) if this applies to the feature;
- To identify the geometrical extent of the feature at any given time; this may change through time as explained in section 5.5.3 and figure 5.20;
- To identify whether the temporal aspect of the 'AccessRestriction' is fulfilled.

5.7.1.3 Geometry

Multiple layers of ground-surface topography are provided by topologically-connected geometrical primitives, from which a 3D geometry can be generated. This 3D geometry allows step heights and ground-surface gradients to be resolved, both of which affect pedestrian access. The pedestrian's 'MaxStep' attribute is assessed against these. Additionally some of the feature types have attributes which describe aspects of their geometry; namely 'Height' attribute for the 'Wall' and 'Door/Window' entities and the 'BaseHeight' attribute for 'Door/Window' entities. These are required by the 'AccessResolver' entity for assessing access.

5.7.1.4 Features

Features have sets of attributes which affect access.

The 'BarrierStrength' attribute indicates the physical ability of the barrier to prevent access to a pedestrian when it is not allowed. Table 5.2 shows the non-linear

Table 5.2: The 'BarrierStrength' ordinal scale.

Value	Term	Meaning
0	freeOpening	A non-physical indication that access is barred through signage or convention; e.g. a 'keep off the grass' sign.
1	indicative	A physical indication that access is barred which does little to physically bar access; e.g. queuing tape in a Post Office.
2	forceableNoDamage	Access barred by a barrier which can be passed if forced, but without causing damage; e.g. a closed and unlocked door.
3	forceableDamage	Access barred by a barrier which, if passed, would result in its permanent damage; e.g. a closed and locked door.
4	secure	A barrier which cannot be passed; e.g. the door to a secure vault.

ordinal scale used in the implementation. Although this is rather subjective, it is intended to distinguish between the willingness and ability to pass a barrier. A burglar is likely to be willing to breach barrier with higher value than a non-burglar in a normal scenario, however a burglar may not have the means to do this (willingness and capability are both represented by the pedestrian's 'MaxBreachLevel'). Some may prefer not to risk the potentially more serious implications of attempting to breach 'forceableDamage' barriers preferring instead to breach only up to 'forceableNoDamage' barriers. In addition to the character of the pedestrian, it is also dependent on context. In an emergency scenario, a pedestrian may breach barriers which they would not normally breach, if required and if they have the means.

The 'AccessPermission' attributes contain most of the detailed accessibility information. They are a set of pedestrian-dependent access permissions which can be attached to certain features ('Space', 'Lift', 'BoundedSpace' and 'Door/Window'). Each access permission is valid for a specific (cyclic) time range.

The pedestrian-dependent permission is in the form of a structured logical expression. The notation used here has individual expressions in square brackets which are combined with 'and', 'or' and 'not' operators, nested using parentheses. Individual expressions in square brackets can be:

- [needs *key ID*] – specifies a key (or other piece of authorisation) which is needed to gain access to the feature.
- [oneway *primitive ID*] – indicates that access must be from one side of the feature, indicated by the ID of the geometrical primitive from which access is gained (when combined with 'and' makes the sub-expression apply in one direction).
- [*attribute operator value*] – an expression which is dependent on the value of an attribute using operators such as '=', '<', '<=', '>' or '>='); an example is '[age = 18]'.

An example is

([age > 18] and ([needs key4356] or [needs key3457]))

Each expression combination can be applied to a cyclic time-range, for example:

weekday, 09:00-17:00

Taken together, an example of a set of access permissions on the main door of an office might be:

weekday 09:00-17:00
weekday 17:00-09:00, [needs key1]
weekend 0:00-24:00, [needs key1]

Access is granted if any of the access permissions in the list is fulfilled. In this case, anyone is allowed access during the weekday working hours between 09:00 and 17:00. Outside of these hours, the pedestrian needs to be in possession of Key1. Access permissions only apply to authorised access; unauthorised access would be granted if the 'MaxBreachLevel' of the pedestrian was high and the security of the door was low.

5.8 Summary

This chapter presented the conceptual model and some of the strategy for implementation. It has defined, justified and explained a set of clear concepts (expressed as entities) which represent a formalisation of a model which can address the design issues expressed in chapter 3. These entities and the relationships between them are used to build an implementation of the data model.

Chapter 6

Implementation

Chapter 5 described an implementation-independent model of the concepts of the (real-world) system being modelled. This chapter will describe the implementation – the logical model and other implementation issues. These implementation issues include the preparation and importing of raw data, caching to improve performance, algorithms for delineating pedestrian access, and resolving a 3D geometry and data output.

The first part of this chapter is concerned with the background and implementation of this specific prototype. Section 6.5 onwards describes the classes with which the concepts of the conceptual model are modelled, illustrating them with UML diagrams. Although this latter part describes classes specific to this implementation, the section has a wider applicability because of the correspondence of the classes to the concepts of the conceptual model and the universal implementation issues addressed.

6.1 Relationship of the conceptual model and the logical model

The model being developed is to be a data repository. However, as the definitions of the 'MappingFramework', the '3DReasoner' and the 'AccessResolver' entities in chapter 5 show, the sole use of a database for the implementation is not enough. A software application is additionally needed to structure the data, maintain the data and generate the outputs required.

Section 4.3.5 saw the use of object-oriented principles (section 4.3) for conceptualising systems and system design. Key to the object-oriented methodology is that components (objects) have *states* and *behaviours*, can operate autonomously or semi-autonomously, and can interact with each other. Chapter 4 also showed that object-oriented principles (section 4.3) are the basis for some programming languages and some databases, which use object-oriented data structures for implementing systems.

There are implementational reasons why using an object-oriented methodology might not be used for implementation. Object-oriented programming has more overhead and complexity than procedural programming. It may need to interface with software libraries that do not fit well into the object-oriented paradigm. Object-oriented databases are still an immature technology, there are few standards, no standard query language and they tend to have poorer performance than other databases (Worboys and Duckham, 2004, p80). A more common approach to developing a complex data-intensive applications is to use an object-oriented programming language and a relational database. This results in a very different logical model for the (object-oriented) software application and the (relational) database.

In the previous chapter and in this chapter, object-oriented principles are used throughout. This is to maintain a focus on the conceptual modelling; prototype development was not an end in itself, but rather a tool to assist in design and for proof-of-concept. The use of the object-oriented principles throughout may be not the best solution from a performance point of view, but this is outside the scope of this study. The implementation uses the Java (section 6.2) programming environment and the object-oriented database, Ozone (section 6.3).

6.2 Programming language: Java

Object-oriented programming languages are designed around object-oriented (OO) principles. The majority of software applications are now written using object-oriented programming languages or procedural languages with object-oriented extensions. Java is an example of an object-oriented programming language.

Java is a strongly typed, high level, object-oriented, platform-independent programming language, with a syntax similar to that of C (Flanagan, 1996). Unlike C++ (C with OO extensions) it has been designed as a pure object-oriented programming language (though it does contain some non-OO concepts such as primitive data types for reasons of runtime efficiency).

Fairly standard simple primitive data types (Boolean, char, byte, short, long, float and double) are provided (Flanagan, 1996, p22). More complicated data types (including text strings) are provided by *classes* by a wide range of standard (distributed with runtime environments by default) and non-standard *classes* (section 6.2.1). User-defined classes can be written and behaviours of existing classes can be modified by defining new classes based on existing classes (extending, inheritance and overriding).

The object-oriented principles described in section 4.3 apply to Java. Data and methods are organised into *classes* which are in a class hierarchy with the common superclass of 'Object'. Classes can roughly correspond to tangible real-world concepts (from the conceptual model); however, classes also encapsulate less tangible concepts such as a 'query' and implementation-specific classes such as 'files' or 'file

writers'. Interfaces are used to define a set of behaviours, and to allow classes in different class hierarchies to be treated with equivalence for these behaviours (an important concept for the object-oriented database used; section 6.3). Classes are additionally organised into *packages* containing classes of related function.

When a new instance of a class is created (an object), it resides in memory and is accessed by a *reference* called a 'variable'. Variables are references to objects in memory (with the exception of primitive data types). Such variables can be easily passed between methods and classes as parameters *by reference*, without the need to send an entire object.

6.2.1 Standard Java libraries

Java is supported by a vast number of standard libraries containing various useful classes. In particular, the classes 'Object', 'String', 'Date', 'ArrayList' and 'HashMap' are widely used. Documentation for these classes can be found at http://java.sun.com/j2se/1.4.2/docs/api/.

The 'java.lang' package includes 'Object' and 'String'. 'Object' is the ultimate superclass of all classes; thus its methods are inherited by all classes (all of which can be overridden). 'String' represents a text string. The 'java.util' package includes the classes: 'Date', 'ArrayList', and 'HashMap'. 'Date' represents the date and time (with millisecond precision). An 'ArrayList' maintains an ordered list of any number of objects (within the constraints of available memory). A 'HashMap' maintains an indexed lists of mappings between an object representing an indexed key and another object representing a value.

6.2.2 Non standard Java libraries

Java's popularity ensures that there are many other useful libraries available, written by various people and organisations. These include Ozone[1] (the object-oriented database used in the implementation), GeoTools[2] (GIS functionality) and the Java Topology Suite[3] (geometrical validation and topological operations).

6.2.3 Cross-platform

An interesting feature of Java is that its compiled classes can be run on any platform for which a Java runtime environment is available. Instead of Java source code being fully compiled, it is translated into platform-independent *bytecode*. These bytecode Java classes run in a *Java Virtual Machine (JVM)* on any platform for which one is available (provided the Java version is compatible).

[1] http://sourceforge.net/projects/ozone/
[2] http://geotools.codehaus.org/
[3] http://www.vividsolutions.com/jts/jtshome.htm

6.3 The database: Ozone

Relational databases (section 4.2.1.1) are the type of database in most common use. Data are input and retrieved using queries written in the built-in query language SQL. The software application must generate SQL to interact with the data.

Object-oriented databases use a different approach. Ozone is an open-source, Java-based object-oriented database and it allows data to be stored persistently as classes, which are easily retrieved by an application written in Java. User-defined classes in Java which are subclasses Ozone's 'OzoneObject' class, can be written and retrieved seamlessly from Ozone. There is no built-in querying language as is the case for relational databases; this must all be done using standard Java programming within the database classes.

Ozone and its user guide (Braeutigam *et al.*, 2003) can be obtained from http://sourceforge.net/projects/ozone/.

6.3.1 Ozone architecture

Figure 6.1 shows the client-server architecture of Ozone. Each class whose instances need to be stored in the database, requires one interface and two supporting classes. The supporting classes are generated automatically by the 'Ozone Post-Processor (OPP)' tool which comes with Ozone. The following classes and interfaces are needed for each class (the last two are generated by OPP):

- The *interface* is used to *refer to a database object or its proxy.* It defines the public methods of the database class and marks those which are not read-only. It is a subinterface of the OzoneRemote class. It is written by the user.
- The *database class* implements the interfaces and it contains all the implemention (code) of the class. It is written by the user.
- The *proxy class* acts as a 'remote control' for its corresponding database class instances. It is automatically generated by OPP.
- The *factory class* handles the creation of the database object. It is used internally by Ozone and further consideration of it is not required. It is automatically generated by OPP.

The Ozone architecture is based on a client-server model. The application (written in Java) is on the *client.* Ozone (also written in Java) runs on the *server.* Database objects reside in Ozone on the server, whereas proxies of these reside in the application on the client. Both these objects can be handled with equivalence, using the interface which they both implement.

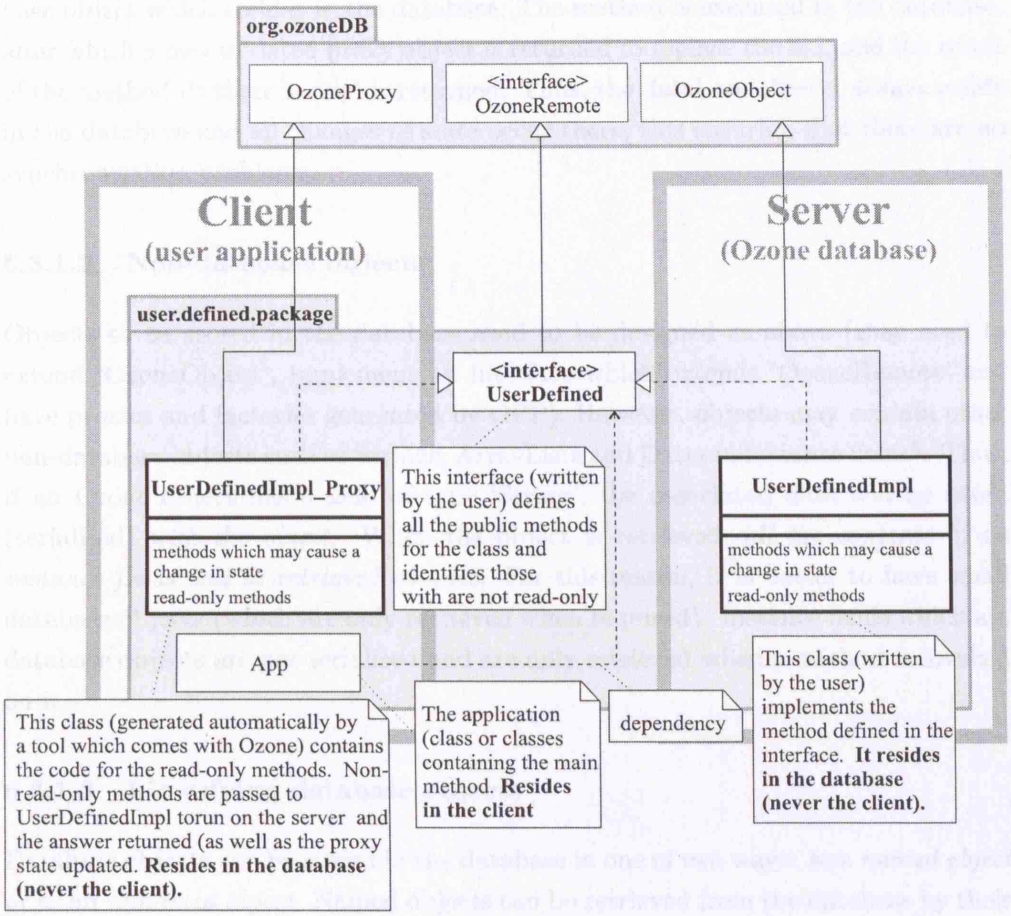


Figure 6.1: The Ozone architecture is based on a client-server model (where the application runs in the client and the database runs on the server). The example shows one user-defined class, instances of which are stored in the database. Each user-defined class requires an interface ('UserDefined'). The interface – which is a subinterface of OzoneRemote – defines all the public methods and marks those which are not read-only. The implementation is written in a class suffixed by 'Impl' ('UserDefinedImpl') and instances of this class reside on the server side. A post-processor which comes with Ozone produces proxy classes suffixed by '_Proxy' ('UserDefinedImpl_Proxy'), instances of which reside on the client.

6.3.1.1 Database objects and proxy objects

When the client retrieves an object (through its interface), it will actually get a *proxy object*, representing the database object. The proxy object appears to function exactly as the database object it replaces. Indeed it does so for methods which are read-only (those that will not result in a change in state of the object); however, methods which may change the state of the object, are passed through to the database object which resides in the database. The method is executed in the database, after which a new updated proxy object is returned to replace the old, and the result of the method (if there is one) is returned. Thus, the database objects always reside in the database and all changes of state occur there; this ensuring that there are no synchronisation problems.

6.3.1.2 Non-database objects

Objects to be stored in the database need to be designed as above (they need to extend 'OzoneObject', implement an interface which extends 'OzoneRemote' and have proxies and factories generated by OPP). However, objects may *contain* other non-database objects such as Strings, ArrayLists and Dates in instance fields[4]. Thus, if an Ozone object has a field of type 'String', the associated data will be saved (serialised) *with the object.* When the object is retrieved, *all the contents of its instance fields will be retrieved at once.* For this reason, it is better to have small database objects (which are only retrieved when required). Instance fields which are database objects are not serialised and are only retrieved when a method is invoked on it.

6.3.1.3 Identifying database objects

Database objects can be stored in the database in one of two ways: as a *named object* or as an *unnamed object.* Named objects can be retrieved from the database by their unique name (an instance of the 'String' class). Unnamed objects must be retrieved through a named object (through a field or method). In this implementation, the database only has one named object: an instance of the 'DataAccessImpl' (equivalent to the 'MappingFramework' entity) which coordinates all the other database objects (section 6.6); all other objects are retrieved through this.

6.4 Implementing the conceptual model

Figure 6.2 shows the classes used in the implementation. These are organised into packages which are used to group the classes according to function. There are many more classes in the implementation than entities in the conceptual model and there

[4]Such object must implement the 'Serializable' Java interface; many do.

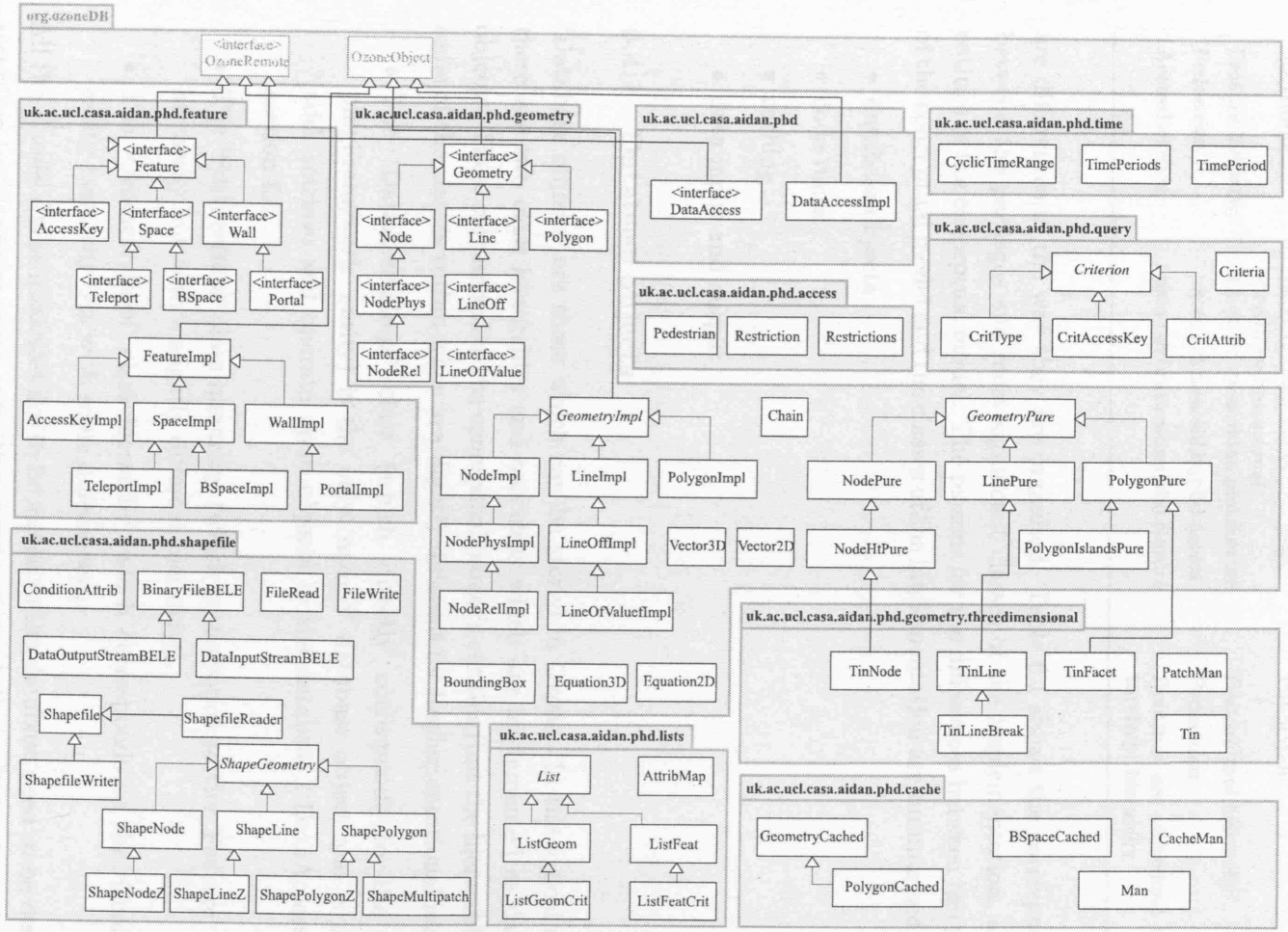


Figure 6.2: An overview of the interfaces and classes of the logical model (with the exception of the classes and interfaces provided by the standard Java packages), organised into packages.

Table 6.1: The correspondence between the entities of the conceptual model (figure 5.1) and the packages and classes of the logical model (figure 6.2). Packages group the classes by related function.

Entity	Package	Class
MappingFramework	uk.ac.ucl.casa.aidan.phd	DataAccessImpl
Geometry hierarchy	uk.ac.ucl.casa.aidan.phd.geometry	GeometryImpl hierarchy
3DReasoner	uk.ac.ucl.casa.aidan.phd.geometry.threedimensional	PatchMan
Feature hierarchy	uk.ac.ucl.casa.aidan.phd.feature	FeatureImpl hierarchy
Pedestrian	uk.ac.ucl.casa.aidan.phd.access	Pedestrian
AccessResolver	uk.ac.ucl.casa.aidan.phd.feature	Routines are embedded in the FeatureImpl hierarchy

are differences in the way they are organised. Table 6.1 shows the correspondence between the packages and most significant classes of the implementation, and the entities of the conceptual model. The reasons for the differences between the entities of the conceptual model and the classes of the implementation are summarised below.

- database objects
- code reuse
- caching
- data input and output

6.4.1 Database objects

'Database objects' are those which can be stored in Ozone. In the logical model, there are two class hierarchies and one class which are implemented as database objects. These correspond to the conceptual model's entities (but the hierarchies and some of the classes within when are slightly different for implementational reasons).

- The 'DataAccessImpl' class (which directly corresponds to the 'MappingFramework' entity) is the only 'named' database object and is able to add, retrieve and operate upon objects in the database ('DataAccessImpl'; section 6.6).
- The 'FeatureImpl' class hierarchy (which corresponds to the 'Feature' entity hierarchy but with a slightly different hierarchy).
- The 'GeometryImpl' class hierarchy (which corresponds to the 'Geometry' entity hierarchy but with some differences).

All the classes whose instances are to be stored in the database need to be designed according to Ozone's requirements (they need to extend 'OzoneObject', need a class name which ends in 'Impl', need to implement a sub-interface of 'OzoneRemote' with the same name as the class but without the 'Impl' and need the supporting factory and proxy classes). All other classes in the implementation are standard

Java classes. Any of these to be *contained* within a database class must implement Java's standard 'Serializable[5]' interface.

6.4.2 Code reuse

In an object-oriented model, one of the criteria for defining classes is to allow for code to be reused. An entity from the conceptual model might be split into two or more classes, one of which would *contain* the others. This allows the contained classes which might have to more universal application to be contained (thus shared) by other classes.

There are many examples of this. Figure 6.2 shows all the classes in the implementation – many more than entities in the conceptual model. In the 'uk.ac.ucl.casa.aidan.phd.access' package, as well as the 'Pedestrian' class (which corresponds to the 'Pedestrian' entity), there are classes called 'Restriction' and 'Restrictions' which do not correspond to any entities. In fact, these correspond to the complicated 'AccessPermissions' attribute present in some of the feature types, where 'Restrictions' is a class which represents a set of individual instances of the 'Restriction' class. Because the access permission are quite complex and are used by multiple classes, it has been packaged up as its own class which can be reused where required.

A much more complicated rationale exists with respect to classes which represent geometry types. In the logical model, classes representing geometries are used in two contexts:

- the conceptual model's 'Geometry' entity hierarchy;
- internally for the temporary and dynamic generation of triangular irregular networks (for modelling the surface geometry of patches) and temporary instances of geometries (e.g. for using the line intersection algorithm).

For this reason, the core geometry functionality has been implemented as a standard Java class hierarchy (all of which are suffixed by the word 'Pure'), topped by the abstract class 'GeometryPure'. These class deal *only with the pure geometrical functionality of the geometry.* The objects of the 'GeometryImpl' database object hierarchy act as 'wrappers', which wrap the corresponding standard Java geometry classes (in the 'GeometryPure' object hierarchy), allowing them to be stored in the database *inside* the database classes. The 'GeometryPure' and 'GeometryImpl' classes all implement the same interfaces can be treated with equivalence. Methods implemented in the GeometryPure classes (those to do with pure geometries, e.g. methods to retrieve the polygon to the left of a line) are invoked by the method of the same name in the GeometryImpl class and vice versa. The relationship between these classes and interfaces is shown in figure 6.20. Although this design appears

[5]http://java.sun.com/j2se/1.4.2/docs/api/java/io/Serializable.html

quite complicated, the complexity is encapsulated and they are handled though their common interfaces, making their use simple.

6.4.3 Caching

For implementation and performance reasons, some way of caching information is required. Because many of the routines are time-dependent, the cache also needs to be time-dependent, so the caching system can maintain cached objects at multiple timestamps. The class 'Man' ('manager') maintains two types of cache: 'CacheMan' which manages cached geometries and features and 'PatchMan' which manages 'Patches' (section 5.4.2).

6.4.3.1 Geometry

The geometries are held in a topologically-structured geometrical model. As such, they use topological relationships to build their geometry; a time-consuming process. For this reason, a special suite of classes for caching the geometries once they have been reconstructed has been implemented. Clearly, the use of cached geometries assumes that the state of the object does not change; for this reason, methods for turning off the caching and methods for clearing the cache exist. Of the geometries, it is the polygons which most need to have their full geometries cached, because their geometrical reconstruction involves the topological querying of their surrounding points and lines.

6.4.3.2 Features

Of the features, it is 'BSpace' (section 6.12.1.4; which corresponds to the 'BoundedSpace' entity) for which caching is appropriate. This is because its geometrical extent is the result of a potentially time-consuming pedestrian- and time-dependent topological query.

6.4.3.3 Patches

The concept of the 'patch', used internally by the '3DReasoner' entity, was described in section 5.4.2. Since patches are incrementally built in an iterative process, the incomplete versions need to be cached. Also, because the whole process is time-consuming, they are appropriate candidates to be cached because once built, they can be used again. The 'PatchMan' (section 6.11) class coordinates the building and caching of patches.

6.4.4 Data input and output

Any implementation of a data model requires a means to enter and retrieve data. Additionally, the 'incremental updating' design principle (section 3.4) requires that data can be added and modified at different times. Designing and building a user interface for importing and adding spatial data is always a challenge. For the prototype implementation, this is purely a means for importing test data; the majority of the development time was directed at the data modelling. Without spending much development time on building a means to input data, a solution had to be found which enabled the data required to be imported.

The difficulties encountered in finding a solution arose from the geometrical and topological nature of the data. The creation of geometrical data is most easily done with a graphical user interface, which is difficult to write. Also, since the data are represented in a topologically-structured geometrical model, the topological relationships needed to be entered. Writing code to generate this from the geometry is not trivial. Also, a way was needed to identify layers which needed to be topologically connected.

The solution eventually chosen was to use the third-party ESRI ArcGIS tool for the preparation of the data through its proprietary, but open ShapeFile file format (ESRI, 1998). ArcGIS can be scripted with VBA and the data were prepared using such scripts. The resulting procedure, described in section 6.6 is rather clumsy, but adequate for prototype testing.

ArcScene (the 3D component of ArcGIS) is used for graphical output, to inspect the results of 3D model generation. Outputs of these form a significant part of the following chapter.

6.4.4.1 Data input

ArcGIS is used to import OS MasterMap data, or to allow floorplans of buildings to be drawn. A VBA script is used to transform the data, build the required geometrical primitives and the required topology for each layer. The topology generated, is automatically placed in DBF tables (using a VBA script). Since each Shapefile is only allowed to have geometries of one type, each 2D 'layer' is composed of 3 Shapefiles for the points, lines and polygons. A series of additional tables (DBF) are also required, into which user-defined information and cross-references are placed.

Features are described in DBF files, one file for each feature type. Rows correspond to instances of features and columns correspond to the features' attributes. One of the columns is for a comma-separated list of the ID numbers of the geometries which comprise the extent of the feature (though not the feature type corresponding to the 'DoorKey' entity). For the feature-type corresponding to the 'BoundedSpace' entity, these geometries represent the starting geometries from which the pedestrian-dependent access query takes place, rather than all the comprising geometries.

6.4.4.2 Data output

The prototype is able to output shapefiles and tables for viewing and editing in ArcGIS, through the 'DataAccessImpl' class). In addition, it is able to output 3D shapefiles of 3D geometry, interpolated by the 'Patches' and the 'PatchMan' (which corresponds to the '3DReasoner' entity).

6.5 Classes and packages

The classes which were designed and implemented are shown in figure 6.2. They are organised into packages which group the classes by function. The diagram shows class inheritance (where inheritance is not shown, the class extends 'Object', the ultimate superclass of all classes). The correspondence of the packages and classes to the entities of the conceptual model is shown in table 6.1 and the reasons for the differences will be given for each package.

Each package will be considered in turn for the remainder of this chapter. To summarise:

- uk.ac.ucl.casa.aidan.phd (section 6.6) – this contains 'DataAccess' which is equivalent to the 'MappingFramework' entity;
- uk.ac.ucl.casa.aidan.phd.shapefile (section 6.7) – classes to deal with reading from and writing to ESRI Shapefiles;
- uk.ac.ucl.casa.aidan.phd.time (section 6.8) – time-related classes.
- uk.ac.ucl.casa.aidan.phd.lists (section 6.9) – these contain specialised lists for holding sets of geometries and features. They also have built-in timestamping of added and removed items, enabling a historical list for any time to be retrieved.
- uk.ac.ucl.casa.aidan.phd.geometry (section 6.10) – classes related to the geometry of the mapping framework.
- uk.ac.ucl.casa.aidan.phd.feature (section 6.12) – classes encapsulating the feature concepts of the conceptual model (section 5.5).
- uk.ac.ucl.casa.aidan.phd.geometry.threedimensional (section 6.11) – classes related to 3D geometry, including a TIN for the surface interpolation of patches.
- uk.ac.ucl.casa.aidan.phd.query (section 6.13) – classes related to queries.
- uk.ac.ucl.casa.aidan.phd.access (section 6.14) – classes describing the access model (section 5.7).
- uk.ac.ucl.casa.aidan.phd.cache(section 6.15) – classes to deal with caching (section 6.4.3).

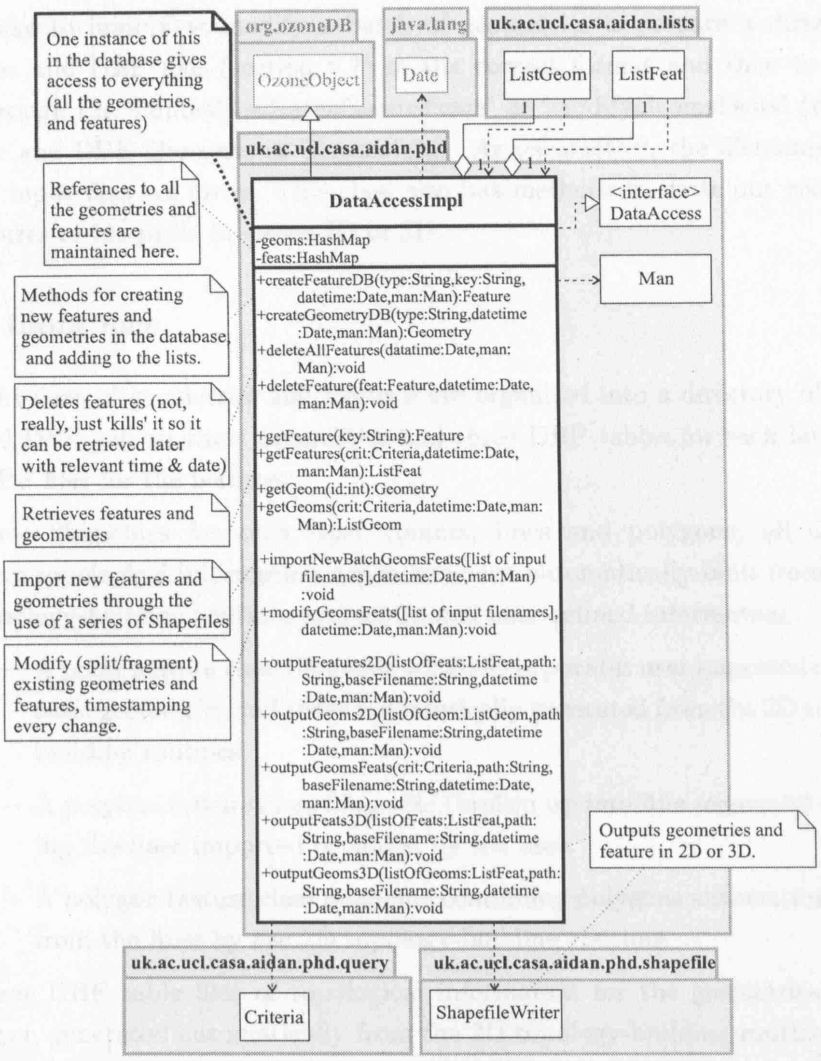


Figure 6.3: UML class diagram of 'DataAccessImpl', showing its fields, methods and dependencies.

6.6 Mapping framework (uk.ac.ucl.casa.aidan.phd)

This package contains the 'DataAccessImpl' class (a database object) and its interface 'DataAccess'. The only 'named object' (section 6.3.1.3) in the database is an instance of this class which is the only point of entry to the data in the mapping framework. This class is directly equivalent to the 'MappingFramework' entity and is responsible for all the operations and queries which need to be performed on the mapping framework, as shown in figure 5.2.

Figure 6.3 shows the class diagram of 'DataAccessImpl'. The method names are quite self-explanatory – there are methods for the input and modification of data and there are methods for the 2D and 3D output of data. Features and geometries can be added individually, using the 'createGeometryDB' and 'createFeatureDB' methods. Features can be retrieved using their 'key' (unique ID) and geometries (which are usually called via features) can be called using their ID number. The

easiest way to import or modify a batch of geometries is prepare a directory of Shapefiles and DBF files (section 6.7) in the correct format and then to import them through the 'importNewBatchGeomsFeats' or 'modifyGeomsFeats' (with the Shapefile and DBF filenames as parameters). As parameters, the filenames of the required input files are given. This class also has methods to write out geometries and features to Shapefile in either 2D or 3D.

6.6.1 Input files

The input data of geometries and features are organised into a directory of ShapeFiles and DBF tables; three ShapeFiles and three DBF tables for each layer, plus seven DBF files for the features:

- three Shapefiles for *each layer* (points, lines and polygons) all of which have topological information which has been automatically-built from the 2D topology-building routines and additional user-defined information.
  - A point feature class Shapefile which incorporates user imported or added node geometries and those automatically generated from the 2D topology-building routines
  - A polyline feature class Shapefile (broken up into line segments) containing the lines imported or added by the user.
  - A polygon feature class Shapefile containing polygons automatically built from the lines by the 2D topology-building routines
- three DBF table files of topological information for the geometries in *each layer*, generated automatically from the 2D topology-building routines
  - a node-line connectivity table
  - a table with information of lines which lie within polygons
  - a table with information about node which lie within polygons which are not otherwise connected to other geometries.
- seven DBF tables representing features *in total*

The filenames for the geometry and topology are prefixed by a number identifying the layer to which they belong. The layers are prepared separately because each needs to have its 2D topology prepared using VBA script in ArcGIS which deals only with 2D topology. When this is done, the places at which layers topologically connect are identified in the attribute tables of the shapefiles (figure 6.7). Other information about the geometries can also be added. The IDs of any features whose extent the geometries form part of, are indicated in the 'feature' column of the Shapefiles' attribute tables (figure 6.7). It is possible to topologically connect with geometry which already exists in the database, in order to fulfil the 'incremental updating' design issue (section 3.4). Although not implemented in the prototype,

it should also be possible to also add more detail into a layer, as was illustrated in figure 5.3.5.

Features are imported as single rows in a set of seven DBF files, the design of which is shown in figure 6.6.

The contents of the directory are interpreted by the 'importNewBatchGeomsFeats' and 'modifyGeomsFeats' methods which import the contents of the directory. When the contents of the directory are imported, the geometry is modelled as a seamless multilayered set of geometrical primitives, which are referenced by the features.

The following steps are required:

- Step 1. Import or draw points (to attach heights) and lines (to bound spaces or be parts of linear features) in separate 2D layers.
- Step 2. Run some VBA scripts which build polygons defined by the lines and build all the 2D topology required by the geometrical model.
- Step 3. Define features (each of the seven feature types is in its own DBF file).
- Step 4. Edit the attribute tables of the geometries to modify the geometrical characteristics and the features which they form part of (see figure 6.7).

6.6.1.1 Step 1 – inputting the geometry

Figure 6.4 shows a screenshot of the Shapefile input for Step 1. Of the Shapefiles loaded, 3 of them have their filename preceded by a '5_' and are 'turned on' (visible). These three shapefiles represent the input nodes, lines and polygons. The outlines of spaces have been drawn (in the '5_input_lines' Shapefile) and a couple of spot heights have been given (in the '5_input_nodes' Shapefile). Polygons are not usually specified.

6.6.1.2 Step 2 – building a topologically-structured layer

Figure 6.5 shows the resulting Shapefiles from running the VBA script. This script builds polygons and nodes from the lines and adds them to the polygons and nodes which were specified in step 1. It allocates them an ID number (in the field 'topol_id') which is used to refer to geometries when specifying topological relationships. In addition to building the polygons (this is done by converting to an ESRI 'coverage' and then exporting the shapes to the shapefiles). The VBA scripts also store the topology required by the geometrical data model in three separate DBF files and within the attribute tables of the geometries:

- '5_newregionline' has the lines and attributes from '5_input_lines', but they have been broken down into individual line segments. The additional attributes 'toNode', 'fromNode', 'leftPoly' and 'rightPoly' have been added, in which the ID of the node or polygon is saved.

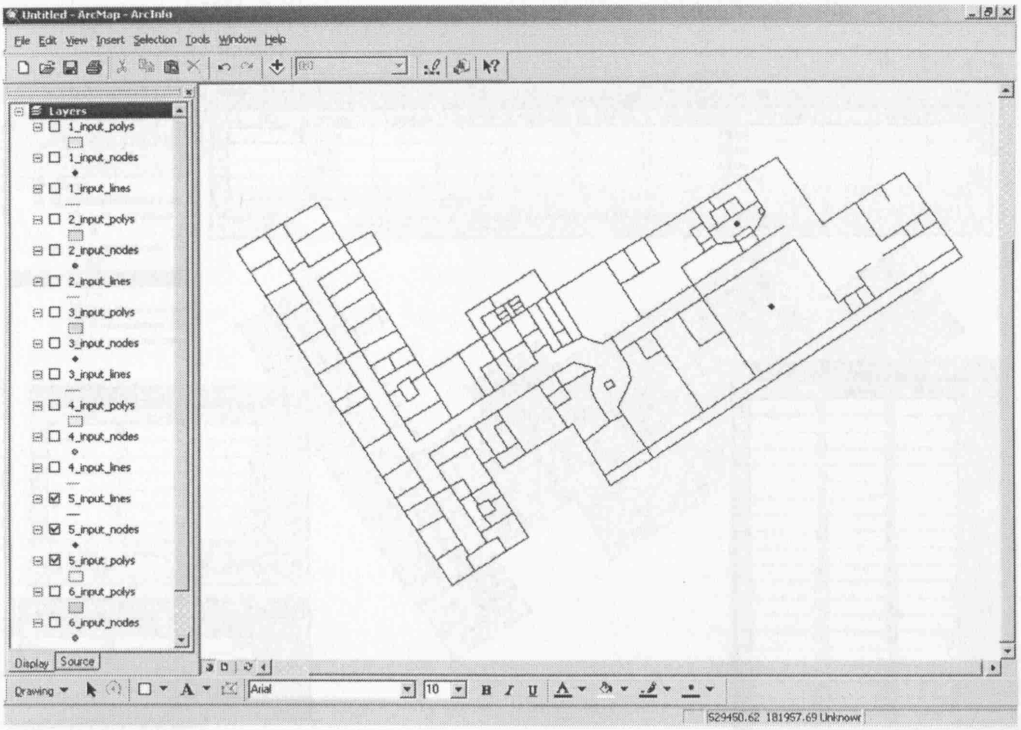


Figure 6.4: Step 1 of the data entry. A screenshot from ArcGIS showing the initial geometry before the VBA topology-building scripts are run. It shows that there are three Shapefiles for each layer (there are no polygons in any of the polygon layers; in the visible layer, there are two nodes and the remainder are lines), layer '5' is 'turned on' (visible) and the Shapefile naming convention for the different layers.

- '5_newregionpoly' has an attribute 'aLine' which stores the ID of any of the line segments around its outer edge.
- The '5_newregionpoly2line' table stores one line from each island for polygons. No entry exists for polygons without islands and multiple entries exist for polygons with more than one island.
- Nodes which are not connected to any lines ('singleton nodes'; section 6.10.5.6) are in this the '5_newregionpoly2node' tables, specifying which polygons they lie within. In this example, there are two (the two spot heights from '5_input_nodes'). Spot heights do not have to be defined in this way – any node can have a height attached.
- The '5_newregionnode2line' table stores the relationship between nodes and the lines which radiate out from them. Nodes connected to three lines will have three entries in the table.

These relationships are read in by the mapping framework are used to populate the geometrical data structures (section 6.10).

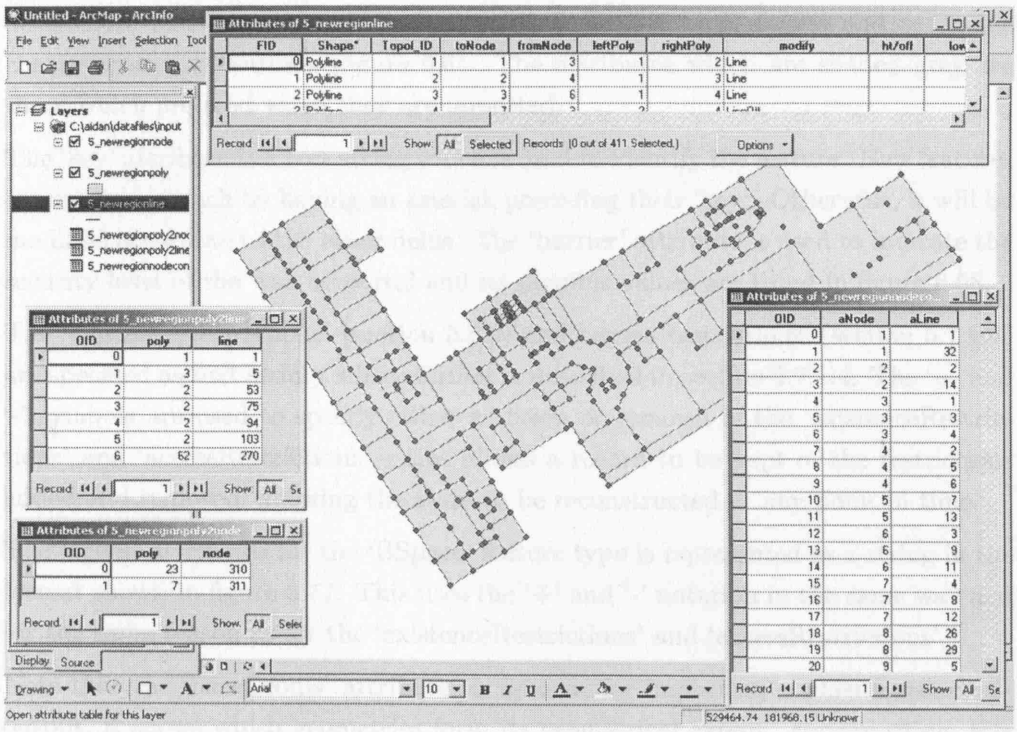


Figure 6.5: Step 2 of the data entry. A screenshot ArcGIS showing the Shapefiles produced by the VBA script for layer 5. Polygons and nodes built from the lines are added to those already specified in step 1. The topology tables are also shown and the extra topology attributes added to the line Shapefile.

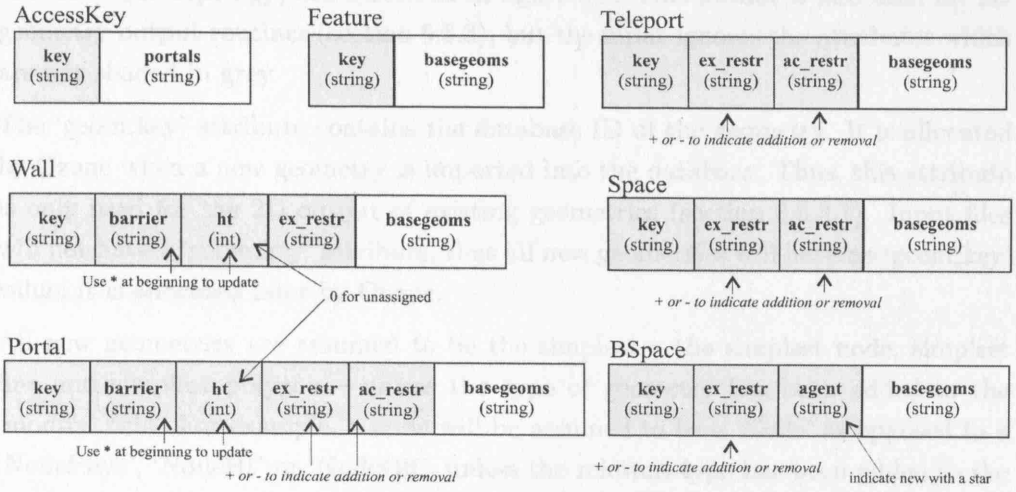


Figure 6.6: The attribute fields for the seven feature types defined by the conceptual model (section 5.5). These are used for input and output. The output routines output all these fields. The input routines only read the shaded fields.

6.6.1.3 Step 3 – defining features

Seven DBF files (one for each feature type) are used to both display and define the characteristics of features (figure 6.6). The attributes which are shaded grey are those which are read when they are imported.

The 'key' attribute is a text string which is used to identify the feature. New features are marked as such by having an asterisk preceding their 'key'. Otherwise, it will be modified according to the other fields. The 'barrier' attribute is used to indicate the security level of the wall or portal and its possible values are listed in figure 6.68.

The 'existenceRestrictions' (section 5.5.3) and 'accessRestrictions' (section 5.7.1.4) are specified as text strings whose format is described in section 5.7.1.4. The '+' and '-' symbols are used to specify either addition or removal of the 'existenceRestrictions' and 'accessRestrictions' – this allows a record to be kept of the restrictions added and removed, allowing the state to be reconstructed at any point in time.

The 'agent' attributes for the 'BSpace' feature type is represented as a string in the format shown in figure 6.77. This uses the '+' and '-' notation in the same way and for the same reason as for the 'existenceRestrictions' and 'accessRestrictions'.

Note that the 'baseGeoms' attributes is ignored for importing – when features are output, it shows which geometries form its geometrical extent. For inputting, this is done through attributes of geometries.

6.6.1.4 Step 4 – editing the attribute tables of geometries

The Shapefile attributes are modified to provide information about the geometries and the features which they define. The Shapefile attribute fields (excluding those related to 2D topology) are described in figure 6.7. This format is also used for 2D geometry output routines (section 6.6.3), but the input ignores the attributes which are not shaded in grey.

The 'geom_key' attribute contains the database ID of the geometry. It is allocated by Ozone when a new geometry is imported into the database. Thus, this attribute is only used for the 2D output of existing geometries (section 6.6.3.1). Input files will not have a 'geom_key' attribute, thus all new geometries will have no 'geom_key' value; it is allocated later by Ozone.

All new geometries are assumed to be the simplest – the simplest node, simplest line and simplest polygon – unless the type of geometry has been added to the 'modify' field. For example, a node will be assumed to be a 'Node' as opposed to a 'NodePhys', 'NodeHt' or 'NodeOff', unless the relevant type has been added to the 'modify' column. The 'ht/off' and 'refgeom' fields will be read if appropriate.

The 'modify' field can also contain other instructions, separated by commas. The 'linkto' keyword is to link a geometry to another. The keyword 'linkto' is followed by either a number, or two numbers separated by an underscore. If there is no

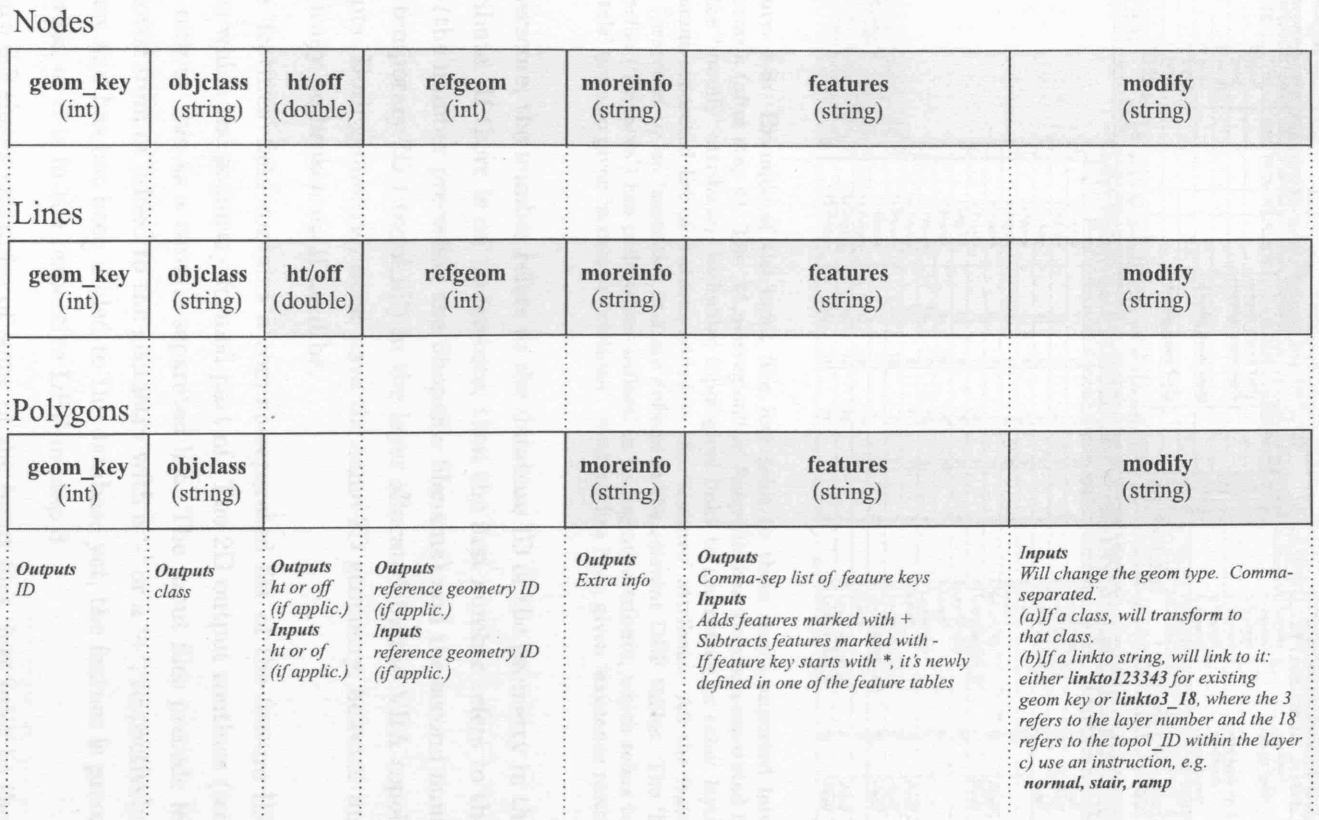


Figure 6.7: The attribute fields for node, line and polygon shapefiles. These are used for input and 2D output. The output routines output all these fields. The input routines only read the shaded fields.

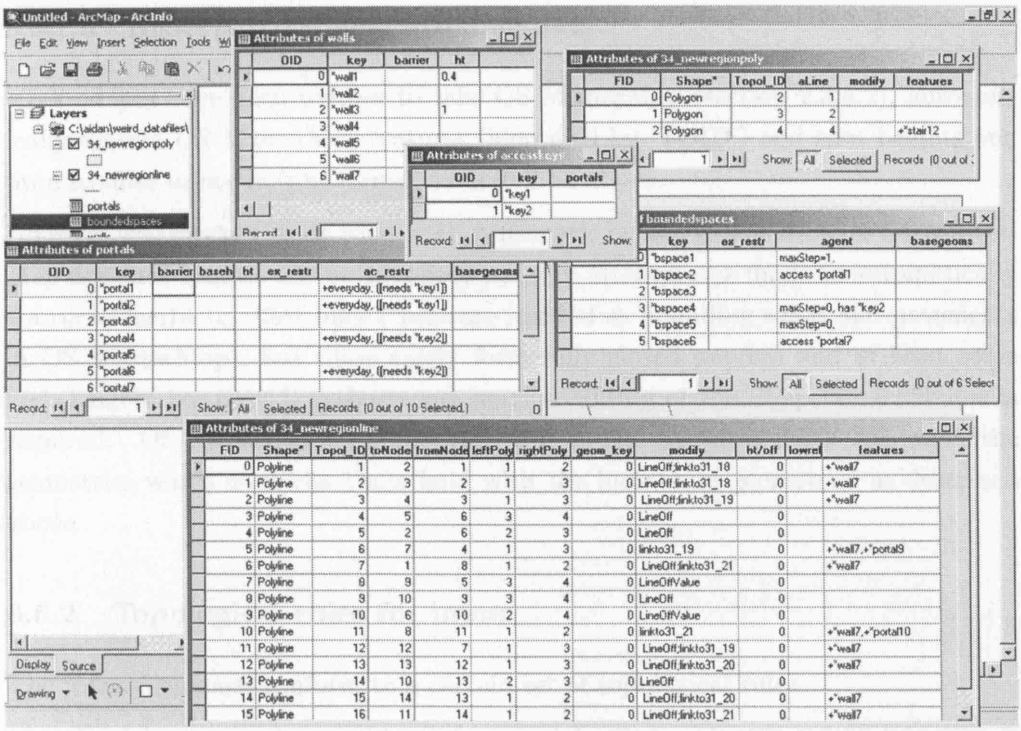


Figure 6.8: Example of the input files just prior to them begin imported into the mapping framework (after step 4). The '34_newregionline' Shapefile has a comma-separated list of modifiers (in the 'modify' attribute), including topological links to geometries in other layers. It also has a comma-separated list of features added in the 'features' attribute. All the features shown are new (preceded by an 'asterisk') and are defined in the relevant DBF tables. The 'boundedspaces' Shapefile ('BSpaces') has pedestrians defined in its 'agent' attribute, which refers to 'AccessKeys'. 'Portals' are also given 'access restrictions'. Nothing has been given 'existence restrictions'.

underscore, the number refers to the database ID of the geometry in the 'geom_key' attribute. If there is an underscore, then the first number refers to the layer number (the number preceding the Shapefile filename) and the second number refers to the temporary ID ('topol_id') in the layer allocated by the VBA topology-building scripts. Both geometries must have the same 2D geometry, however lines will automatically be broken up if need be.

The 'features' field contains a comma-separated list of the feature IDs of the features which the geometry formed part of. The 2D output routines (section 6.6.3.1) will output this as a comma-separated list. The input files precede features being removed from or added to the geometry with a '-' or a '+', respectively. If a feature is new and has not been added to the database yet, the feature is preceded with an asterisk, as it is in the respective DBF in step 3.

Figure 6.8 shows examples of some of the input files, just prior to them being imported into the mapping framework.

6.6.1.5 Importing national mapping data

VBA scripts have been written to take OS MasterMap (section 2.3.5.2), automatically extract OS MasterMap features (identified by 'TOID') and spot heights and produce files in the output format as described above.

However, the process is not completely automatic by any means because OS MasterMap does not contain all the data required. An attempt was made to automatically transform kerbs to offset lines (these are marked as bounding roads and pavements in OS MasterMap), but these rarely form fully closed patches and at least some kerb heights are required; thus some manual editing of the Shapefile attributes is required. OS MasterMap features are held in the 'features' DBF file, with the geometries which comprise them held with the individual geometries as described above.

6.6.2 Topological rules for input

The input files must conform to a certain set of topological rules.

- Each layer must be a valid set of non-overlapping geometries with planar topology enforcement and the required topology must be explicitly described in attribute table and DBF files as described above.
- Layers which should geometrically join must have this described explicitly (though a 'linkto' attribute) in one of the layers (these are automatically assumed to be breaklines or are specified as offset lines).
- Every set of geometries which will form a patch must have at least one piece of height information (heighted nodes within, heighted nodes on the boundary or height offset lines.
- Stairs and ramps are composed of polygons marked as such and the polygon or contiguous set of polygons must be surrounded by offset lines or be at the layer's edge.

6.6.3 Output files

The mapping framework can output in 2D and 3D.

6.6.3.1 2D output

Given a set of geometries, a set of features or a query, the output routines output the geometry as three shapefiles (nodes, lines and polygons, with the attributes) and seven DBF files in the same form as for the input files. Each feature in the DBF file has a comma-separated list of the geometries which comprise its extent in the 'geoms' file. However, since the geometry is stored seamlessly, the original layers as imported are gone, being collapsed into a single layer.

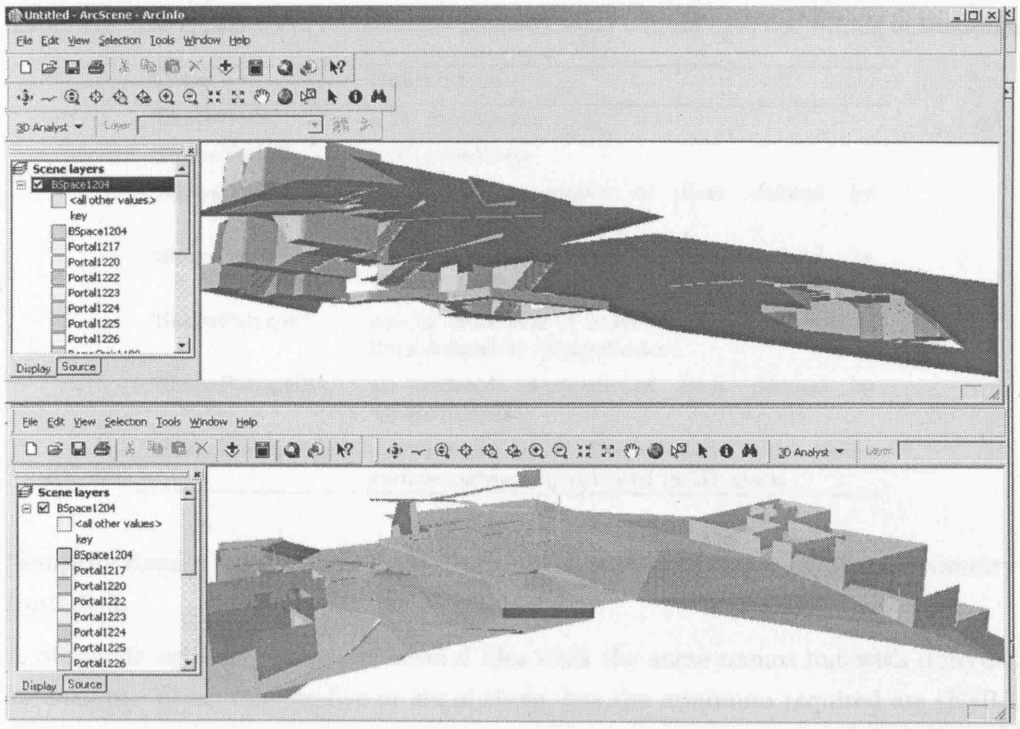


Figure 6.9: Example of a 3D output from two different viewing angles, with the geometries making up different features arbitrarily coloured.

Since the files are in a similar format to the files imported, modifications such as adding and removing access/existence restrictions and changing the spatial extent of features ('using '+' and '-') can be done, and then the file reimported. Only new information (proceeded with '*', '-' or '+') is modified – the other information is ignored.

6.6.3.2 3D output

The 3D output routines build a 3D geometry from the height constraints (section 5.3.4), using the 'PatchMan' class (section 6.11). 3D output is achieved by the output of 3D Shapefiles using the 'ShapeMultiPatch' Shapefile geometry type (section 6.7).

6.7 Shapefile (uk.ac.ucl.casa.aidan.phd.shapefile)

The classes in this package are used by 'DataAccessImpl' to read and write Shapefiles. It is an implementation-specific set of classes which did not feature in the conceptual model.

This package has a collection of classes which deal with ESRI Shapefiles (ESRI, 1998) and DBF files (ESRI, 1998; Borland, 1998). They were developed in accordance with ESRI's technical specification. The details of this classes will not be discussed in detail; it is sufficient to know that the 'ShapefileWriter' class and the

Table 6.2: The classes which represent the geometry types for read from and writing to Shapefiles

Class	Description
'ShapeNode'	a 2D coordinate
'ShapeNodeZ'	a 3D coordinate
'ShapeLine'	an ordered sequence of lines defined by 'ShapeNodes'
'ShapeLineZ'	an ordered sequence of lines defined by 'ShapeNodeZs'
'ShapePolygon'	one or more sets of ordered closed sequences of lines defined by 'ShapeNodes'
'ShapePolygonZ'	an ordered sequence of lines defined by 'ShapeNodeZs'
'ShapeMultipatch'	a structured set of 'ShapeNodeZs' which define surfaces arbitrarily oriented in 3D space.

'ShapefileReader' class respectively write and read subclasses of ShapeGeometry from disk.

A Shapefile actually comprises several files with the same names but with different extensions. There may be five or six of these, but the minimum required are (ESRI, 1998):

- .shp – the geometrical description.
- .shx – an index of the position of each shape in the file
- .dbf – a standard DBF tabular file format for storing attributes of shapes, where rows are records and columns are fields (Borland, 1998)

DBF tables can also exist on their own as tabular data with no additional spatial information.

The Shapefile format can only have one type of geometry per file. The Shapefile geometry types in this package (subclasses of the abstract class 'ShapeGeometry') are shown in table 6.2.

The 'ConditionAttrib' class allows a value to be automatically inserted into a field depending on a 'Criteria' being fulfilled.

6.8 Time (uk.ac.ucl.casa.aidan.phd.time)

Time is an integral part of the framework and is used by most of the classes. In order for these concepts to be reused in all the classes which rely on them, the concepts of snapshot time and cyclic time (section 4.8.1) have been packaged up into their own classes (thus they are not mentioned in the conceptual model). Features use snapshot time to represent their creation and deletion times and the times at which geometries are added and removed (through a timestamped list; section 6.9) and some use cyclic time to represent their existence. The access permissions (section

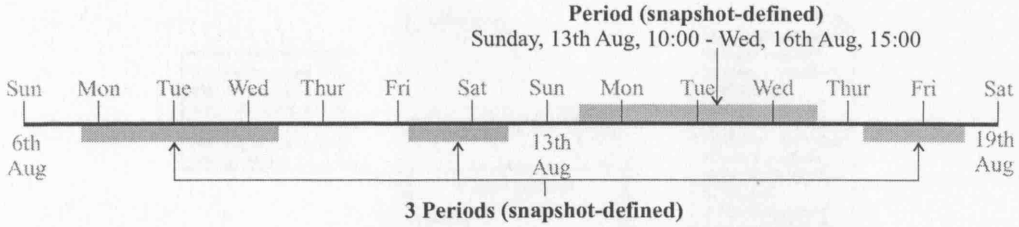


Figure 6.10: Time periods based on 'snapshot time'.

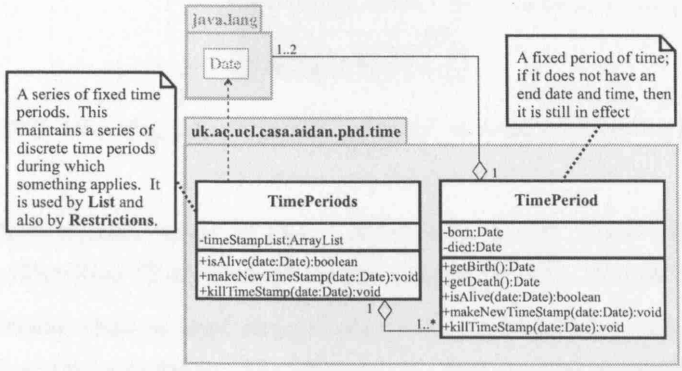


Figure 6.11: UML class diagram of 'TimePeriods' and 'TimePeriod', showing their fields, methods and dependencies.

6.14) on features use cyclic time and any method which resolves access or constructs features for a given time relies on the classes in this package.

Section 4.8 distinguished between the different contexts in which the times of states apply: in the real-world, when noticed, when surveyed and when entered in the database. It acknowledges the likely lag-time between these, providing a way in which it can be managed. In spite of this, the implementation designed only deals with one type of time for the purposes of simplicity. This effectively assumes that all the contexts which time can apply occur at the same instant. The use of time is such databases is evaluated and discussed in section 7.3.4.

6.8.1 Snapshot time

'Snapshot time' represents an instant in time. The standard Java 'java.lang.Date' class is used for to represent this in the implementation. (It so happens to represent time to millisecond precision, but this level of precision is not needed).

As shown in figures 4.22 and 6.10, instants in time can be used to describe time periods and series of time periods. This concept is encapsulated by the 'TimePeriod' and 'TimePeriods' classes respectively, which is shown in figure 6.11.

'TimePeriod' represents one time period between two 'Dates'. It is not used directly by other parts of the implementation; rather it is used through 'TimePeriods' which maintains an ordered list of these, representing a series of time periods. The 'makeNewTimeStamp' and 'killTimeStamp' methods can be given any 'Date'. The

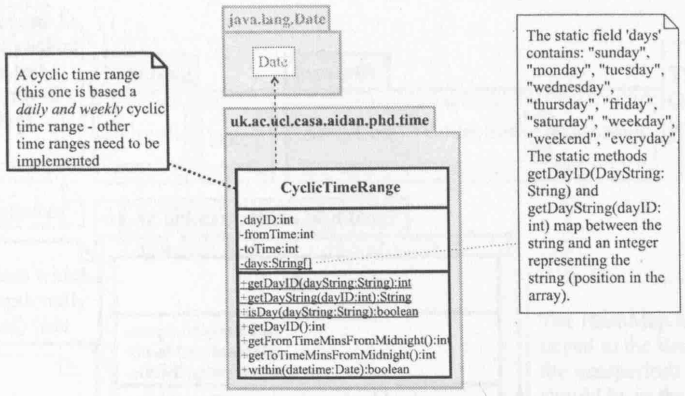


Figure 6.12: UML class diagram of 'CyclicTimeRange', showing its fields, methods and dependencies.

'isAlive' method returns 'true' or 'false' depending on the 'Date' parameter value falls within a described 'TimePeriod' within the list held by 'TimePeriods'.

The 'TimePeriods' class is used extensively. It is used by 'List' and its subclasses, 'FeatureImpl' and its subclasses (the lifecycle of a feature and changes to its geometrical extent through time), and 'Restrictions' (changes to access restrictions through time).

6.8.2 Cyclic time ('CyclicTimeRange')

Cyclic time represents a recurring instant in time and a cyclic time period is a recurring time period. This concept is encapsulated by the 'CyclicTimeRange' class, shown in figure 6.12. The current implementation of the cyclic time range based on a daily and weekly cycle. A single day or group of days and time range can be chosen.

The constructor of 'CyclicTimeRange' (not shown) takes a 'String', which is interpreted to populate the fields 'dayID', 'fromTime' and 'toTime'. The string is in the format as described in section 5.7.1.4.

6.9 Lists (uk.ac.ucl.casa.aidan.phd.lists)

This package contains a hierarchy of classes topped by 'List' which are used internally by the implementation (thus not mentioned in the conceptual model) to maintain lists of geometries and features. They have built-in timestamping; every addition and deletion of geometries or features is timestamped such that the state of the list at an instant in time can be reconstructed, through the 'getAsArrayList' method which takes a 'Date' as a parameter. Some of the classes can also ensure that only certain types of object are added (controlled by an instance of the 'Criteria' class; section 6.13.1). The 'ListGeom' class is used by the classes in the 'FeatureImpl' hierarchy for holding a list of the geometries which comprise their extents. Conversely, the

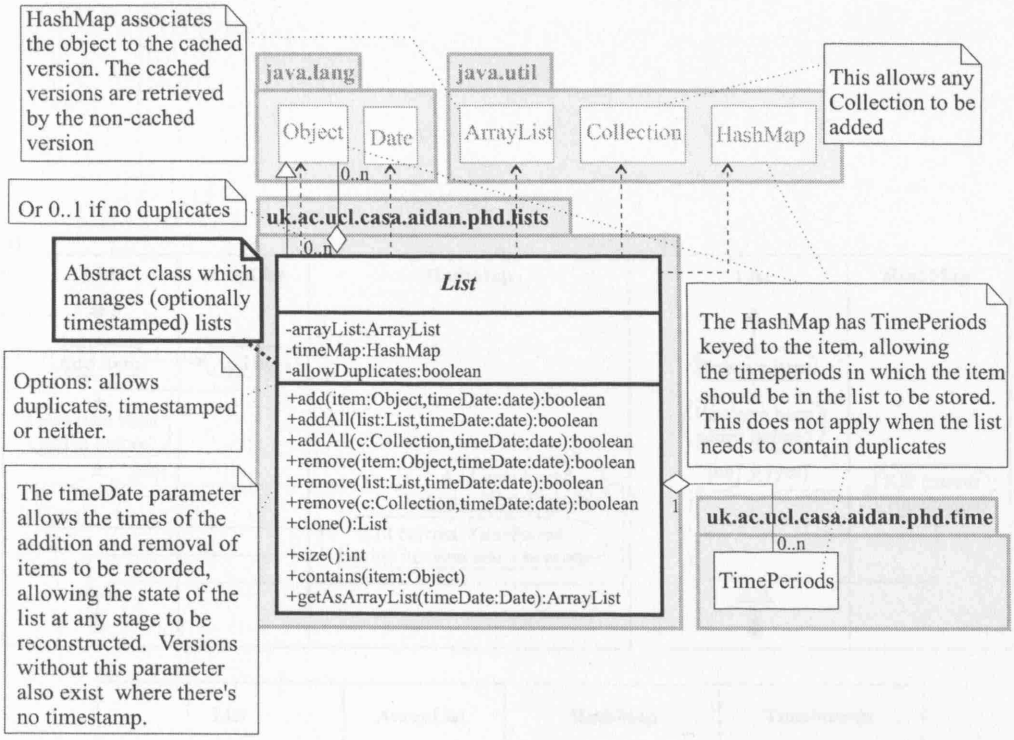


Figure 6.13: UML class diagram of 'List', showing its fields, methods and dependencies.

'ListGeom' class is used by the classes in the 'GeometryImp' hierarchy for holding the features whose extent form parts. This package also contains the 'AttribMap' class which maintains attributes as a list of attributes names with values. This class is used by classes in the 'FeatureImpl' hierarchy for maintaining feature attributes.

6.9.1 'List' and its subclasses

6.9.1.1 'List'

Figure 6.13 shows the class diagram of the abstract class 'List'. Internally, items are stored in the 'arrayList' field and methods are available which allow the list to be modified and information to be retrieved.

Figure 6.14 shows that the methods for adding items, removing items and retrieving items are time dependent. The times at which items are added and removed from the list are recorded. Items are never actually removed because they may need to be retrieved later. The time periods in which the items are valid are managed by the 'TimePeriods' class (section 6.8). A list at any time state can be retrieved.

'List' is an abstract class and this temporal functionality is inherited by 'ListGeom' 'ListGeomCrit', 'ListFeat' and 'ListFeatCrit'.

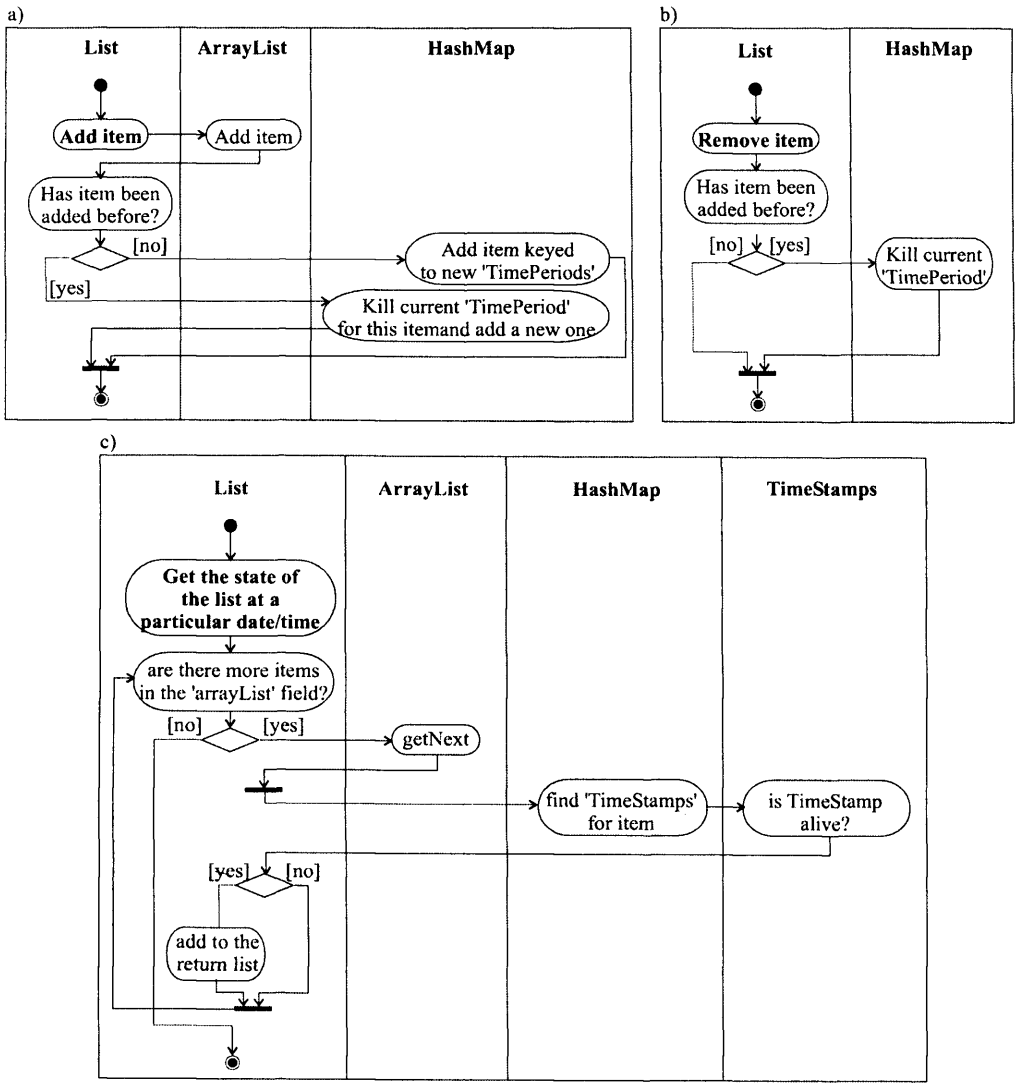


Figure 6.14: UML sequence diagram for 'List'. *a)* When an item is added, it is added to the 'arrayList' (for the order') and added to the 'timeMap' mapping which maps items to 'TimeStamps'. A new 'TimeStamps' is added if the items has never been added before, otherwise, a new TimeStamp is added. *b)* When an item is removed, it is not removed from the 'arrayList', but the associated 'TimeStamps' is 'killed'. *c)* When the state of the list is retrieved at any snapshot, each item (in order) in the 'arrayList' is processed. Any whose 'TimeStamps' are 'alive' are added to a temporary 'ArrayList' which is returned.

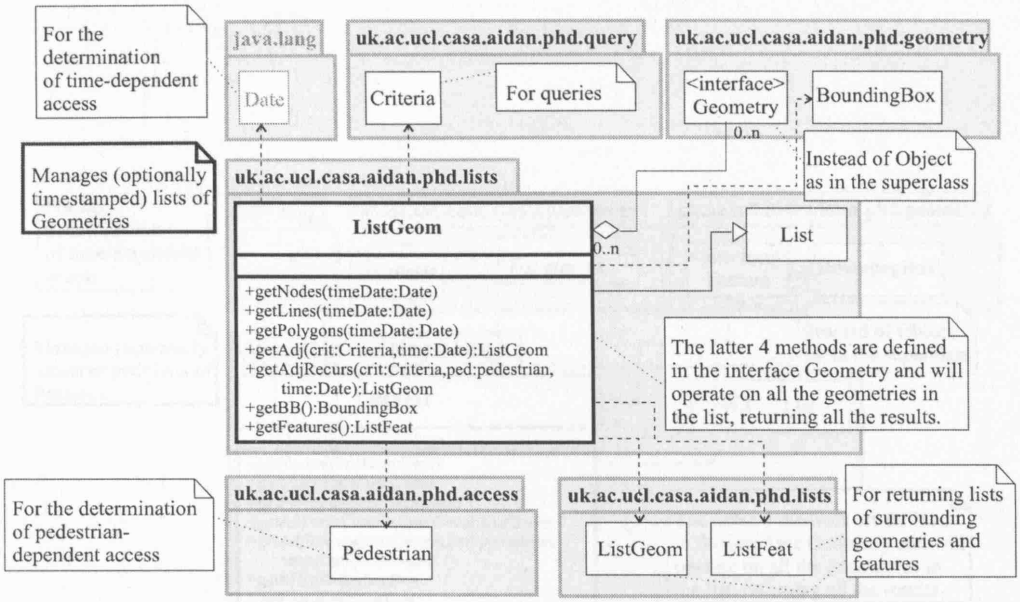


Figure 6.15: UML class diagram of 'ListGeom', showing its fields, methods and dependencies.

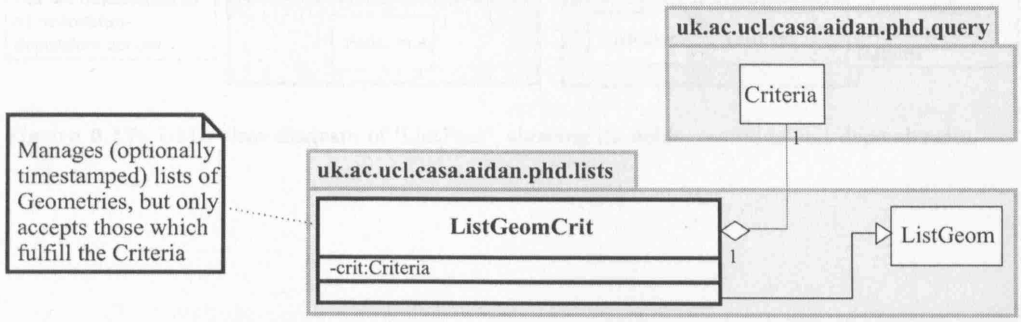


Figure 6.16: UML class diagram of 'ListGeomCrit', showing its fields, methods and dependencies.

6.9.1.2 'ListGeom' and 'ListGeomCrit'

Figure 6.15 shows the class diagram of the class 'ListGeom'; this is as its superclass ('List'), but it only holds geometries (objects which implement 'Geometry'). It has some extra methods over 'List' relating to geometry which are passed through to the all the geometries held in the list. Thus, the functionality of 'getAdj' and 'getAdjRecurs' are described in section describing the 'GeometryPure' class hierarchy (section 6.10.5).

Figure 6.16 shows the class diagram of the class 'ListGeomCrit'. This allows a 'Criteria' (section 6.13.1) to be stored in the list which acts as filter allowing only items which conform to the 'Criteria' to be added.

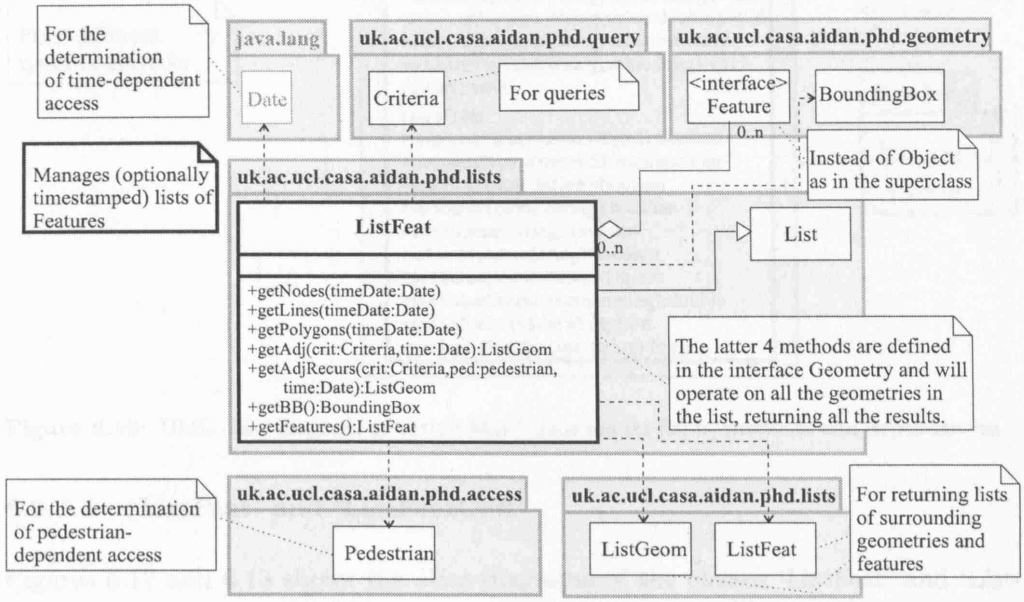


Figure 6.17: UML class diagram of 'ListFeat', showing its fields, methods and dependencies.

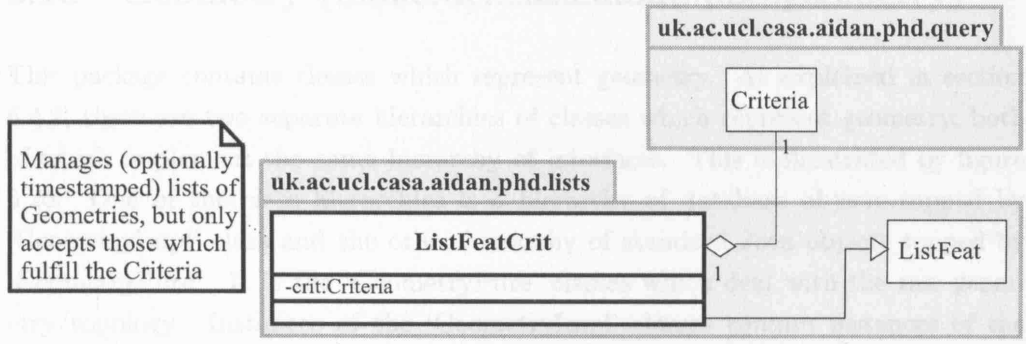


Figure 6.18: UML class diagram of 'ListFeatCrit', showing its fields, methods and dependencies.

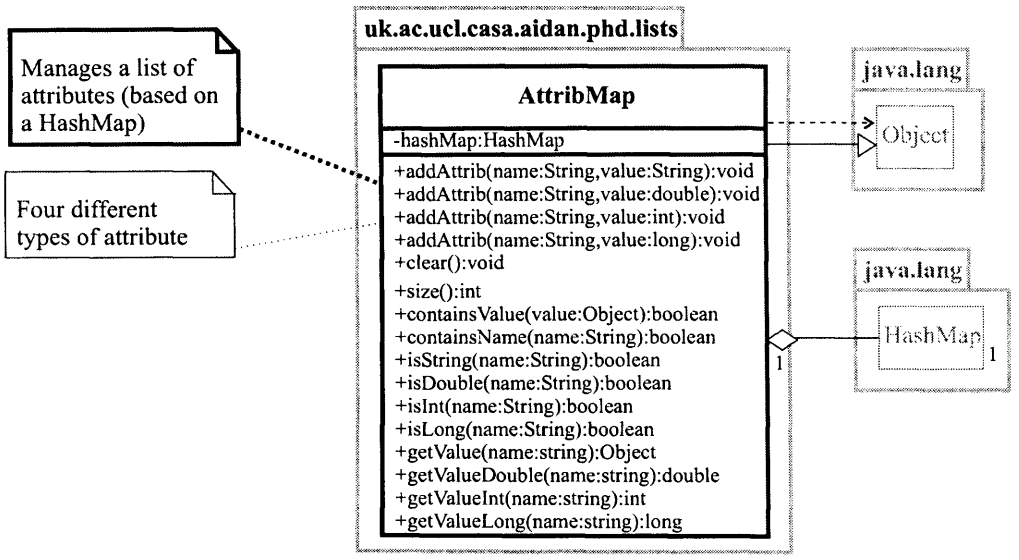


Figure 6.19: UML class diagram of 'AttribMap', showing its fields, methods and dependencies.

6.9.1.3 'ListFeat' and 'ListFeatCrit'

Figures 6.17 and 6.18 shows the class diagrams of the classes 'ListFeat' and 'ListFeatCrit'. These are very similar to 'ListGeom' and 'ListGeomCrit', except that they apply to features (objects which implement 'Feature').

6.9.1.4 'AttribMap'

Figure 6.19 shows the class diagram of the class 'AttribMap', which maintains a list of attributes. It is implemented as a HashMap with methods to support the reading and writing of four types of attribute: text string, double, integer and long.

6.10 Geometry (uk.ac.ucl.casa.aidan.phd.geometry)

This package contains classes which represent geometry. As explained in section 6.4.2, there are two separate hierarchies of classes which represent geometry, both of which implement the same hierarchy of interfaces. This is illustrated by figure 6.20. One of the class hierarchies is a hierarchy of database objects topped by 'GeometryImpl' class and the other hierarchy of standard Java objects topped by 'GeometryPure'. It is the 'GeometryPure' classes which deal with the raw geometry/topology. Instances of the 'GeometryImpl' classes *contain* instances of the 'GeometryPure' classes.

It is the 'GeometryImpl' hierarchy which corresponds to the 'Geometry' entity hierarchy of the conceptual model. However, there are some differences in the implementation and these are shown in table 6.3. The main differences are:

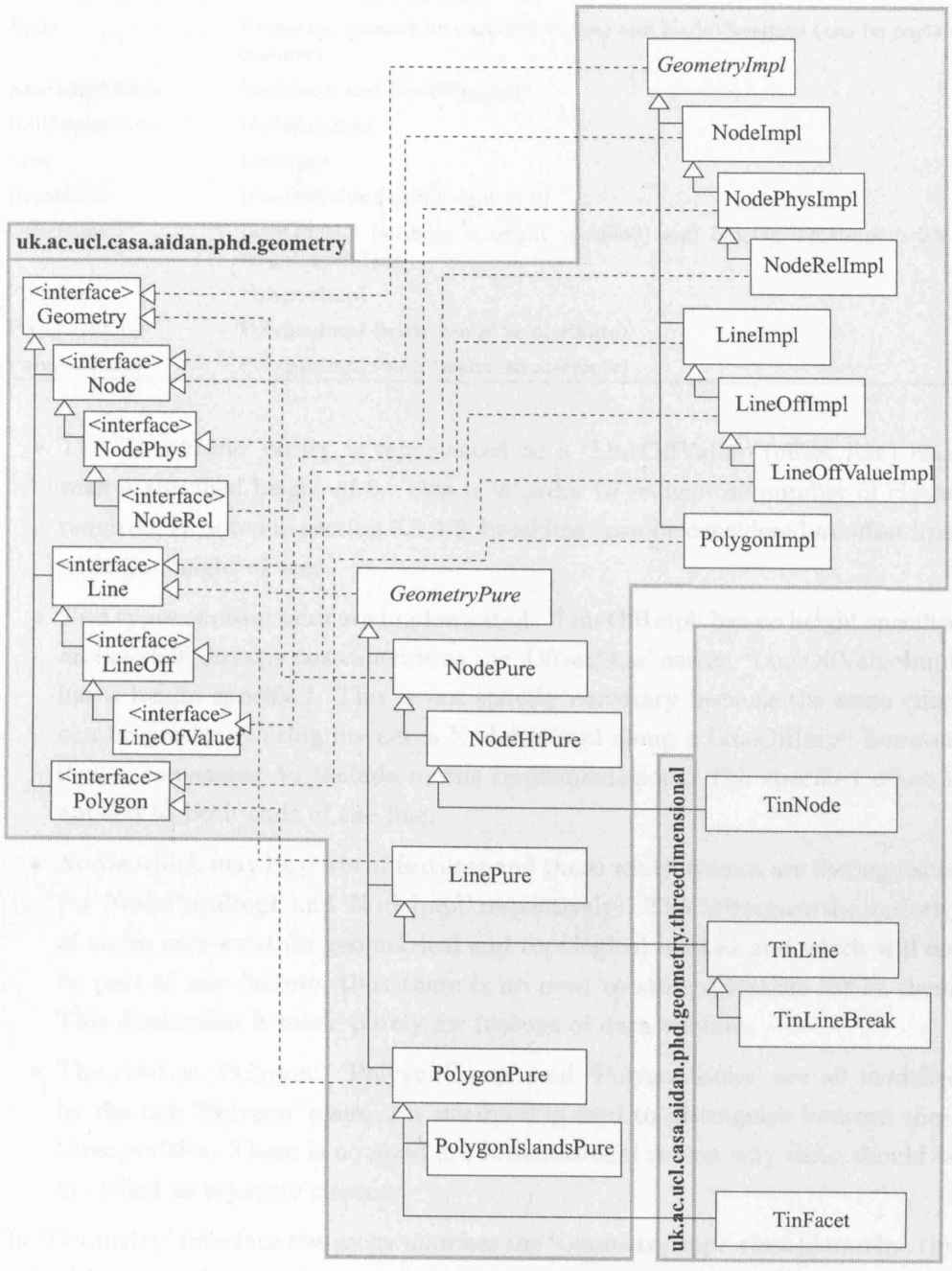


Figure 6.20: UML diagram showing the relationships of the 'GeometryImpl' hierarchy of classes (database objects) and the 'GeometryPure' hierarchy of classes (standard Java classes), with the 'Geometry' hierarchy of interfaces that they implement.

Table 6.3: The correspondence between the 'Geometry' entity hierarchy of the conceptual model and the 'GeometryImpl' class hierarchy of the logical model.

Entity	Class
Geometry (abstract)	GeometryImpl (abstract)
Node	NodeImpl (cannot be part of features) and NodePhysImpl (can be parts of features)
AbsHeightNode	NodeImpl and NodePhysImpl
RelHeightNode	NodeRelImpl
Line	LineImpl
BreakLine	LineOffValue (with height of 0)
OffsetLine	LineOffImpl (without a height specified) and LineOffValueImpl (with a height specified)
Polygon	PolygonImpl
PolygonRamp	PolygonImpl (with 'ramp' as attribute)
PolygonStairs	PolygonImpl (with 'stairs' as attribute)

- The 'BreakLine' entity is represented as a 'LineOffValue' (offset line) class with a specified height of 0. This is in order to reduce the number of classes required; as noted in section 5.3.4.2, breaklines can be considered as offset lines with the height of zero.
- Two types of offset lines are implemented. 'LineOffImpl' has no height specified and as such directly corresponds to the 'OffsetLine' entity. 'LineOffValueImpl' has a height specified. This is not strictly necessary because the same effect can be produced using instances NodeRelImpl along a LineOffImpl; however, it was convenient to include in the implementation. The specified offset is applied to both ends of the line.
- Nodes which may be parts of features and those which cannot are distinguished (as 'NodePhysImpl' and 'NodeImpl' respectively). This is because the majority of nodes only exist for geometrical and topological reasons and which will not be part of any feature, thus there is no need to store a feature list to them. This distinction is made purely for reasons of data volume.
- The entities 'Polygon', 'PolygonRamp' and 'PolygonStairs' are all modelled by the one 'Polygon' class. An attribute is used to distinguish between these three entities. There is no good implementational reason why these should be modelled as separate classes.

The 'Geometry' interface hierarchy matches the 'GeometryImpl' class hierarchy; this is a requirement of Ozone (section 6.3). However, there is not such a correspondence between the interfaces and the 'GeometryPure' class hierarchy, because they deal only with the pure geometry. Instances of 'NodePure' and 'NodeHtPure' deal with 2D and 3D nodes respectively; either can be contained any by objects of 'NodeImpl' or its subclasses. Instances of 'LinePure' are contained by all objects of 'LineImpl' or its subclasses. Instances of 'PolygonPure' and 'PolygonIslandsPure' deal with polygon without and with islands (holes) respectively. Either can be contained by

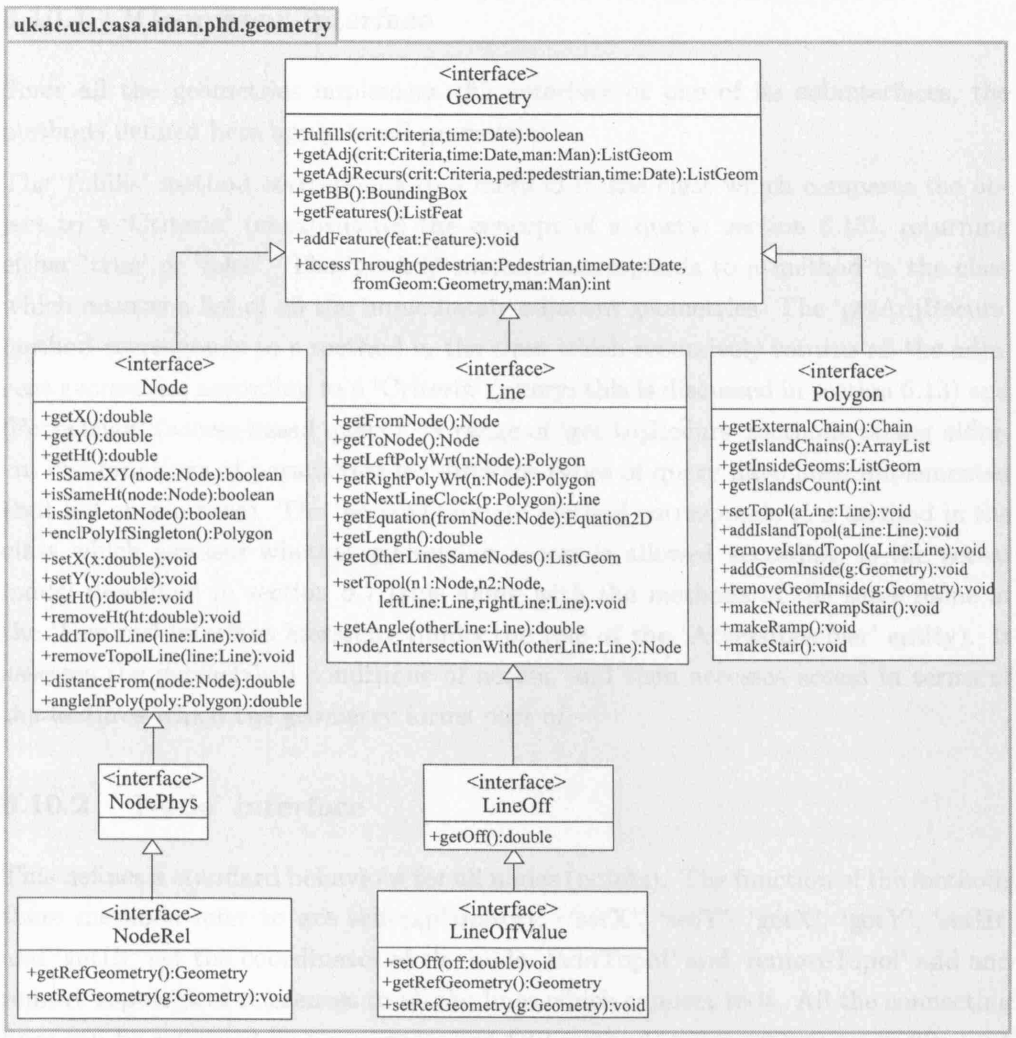


Figure 6.21: UML interface design defining the methods which should be implemented by the 'GeometryImpl' and 'GeometryPure' class hierarchies.

instances of 'PolygonImpl'. Thus, the pure geometrical aspects of nodes (whether they are 2D or 3D) and of polygons (whether they have islands or not) are of no concern to the GeometryImpl class hierarchy.

Figure 6.21 shows method of the 'Geometry' interface hierarchy. These interfaces are implemented by 'GeometryPure' and 'GeometryImpl' (and 'GeometryImpl_Proxy'), so the methods defined can be used for all the geometries.

6.10.1 'Geometry' interface

Since all the geometries implement this interface or one of its subinterfaces, the methods defined here apply to all geometries.

The 'fulfills' method corresponds to a method in the class which compares the object to a 'Criteria' (encapsulates the concept of a query; section 6.13), returning either 'true' or 'false'. The 'getAdj' method corresponds to a method in the class which returns a list of all the immediately adjacent geometries. The 'getAdjRecurs' method corresponds to a method in the class which recursively returns all the adjacent geometries according to a 'Criteria' (query; this is discussed in section 6.13) and 'Pedestrian' (access-based query). A range of 'getAdjRecurs' methods taking different combinations of parameters for different types of query have been implemented (but not shown here). The 'accessThrough' method corresponds to a method in the class which assesses whether pedestrian access is allowed according to the access model described in section 5.7 (this along with the methods of the same name in the 'Feature' interface hierarchy fulfills the role of the 'AccessResolver' entity). It assesses the geometrical conditions of access, and then accesses access in terms of the features which the geometry forms part of.

6.10.2 'Node' interface

This defines a standard behaviour for all nodes (points). The function of the methods these methods refer to are self-explanatory. 'setX', 'setY', 'getX', 'getY', 'setHt' and 'getHt' set the coordinates of the node. 'addTopol' and 'removeTopol' add and remove topological references to all the lines which connect to it. All the connecting lines can be retrieved by Geometry's 'getAdj' method, using a 'Criteria' to filter out the geometries which are not lines.

6.10.3 'Line' interface

This defines a standard behaviour for all lines; setting and retrieving topological information and computing information about itself. The 'getLeftPolyWrt' and 'getRightPolyWrt' methods return the left and right polygon with respect to one of its two nodes. 'getNextLineClock' returns the next line in a clockwise order with respect to one of its bounding polygons – this method is used to retrieve the

geometry of polygons. It can also compute the position of its intersection with another line, returning an object of the 'NodePure' class (temporary and not stored in the database).

6.10.4 'Polygon' interface

This defines a standard behaviour for all polygons. The methods are described in the descriptions of 'PolygonPure' and 'PolygonIslandsPure' (section 6.10.5.1).

6.10.5 'GeometryPure' and its subclasses

The 'GeometryPure' class hierarchy deals with the raw geometry/topology. The geometry is modelled using a 2D topologically-structured data model, similar to that used by the ESRI ArcInfo Coverage model (figure 4.10). This model uses planar enforcement (section 4.5.4.1) and the following topological rules apply:

- a node can have any number of lines connected to it (0..n)
- a line can have up to two bounding polygons (0..2)
- a line is defined by two nodes (they are line segments rather than the line segment sequences shown in figure 4.10)
- a simple polygon is described by one closed chain (sequence of line segments); a complex polygon (with islands) is described by more than one chain. A chain is represented by the 'Chain' class.

Support for 'multilayering' (section 5.3.3) allows different layers to be described. As described in section 5.3.3 and illustrated in figure 5.6, these different layers need to be topologically connected at certain places. However, the topological properties of these places are not 2-manifolds, thus they do not conform to the 2-manifold topological model. The solution to this is to have coincident lines which share nodes and to treat the coincident lines are one line which can have more than two adjacent polygons. Routines such as 'getAdj' which is implemented by the geometries do this, treating the non-2-manifold as topologically seamless.

Figure 6.22 shows the classes in this class hierarchy. 'GeometryPure' is an abstract class; 'NodePure' is a simple 2D node; 'NodeHt' is a 3D node; 'LinePure' is a straight line segment; 'PolygonPure' is a simple polygon with no islands; and 'PolygonIslandsPure' is a polygon containing islands. The other classes in the 'uk.ac.ucl.casa.aidan.phd.geometry.threedimensional' package are considered in section 6.11).

Implementation of geometrical models tend to have precision problems. Very narrow 'sliver' polygons for example can cause problems. The geometrical model in this implementation uses simple (and rather crude) snapping within a particular tolerance. This presents problems where the tolerance is too high for a particular geometry.

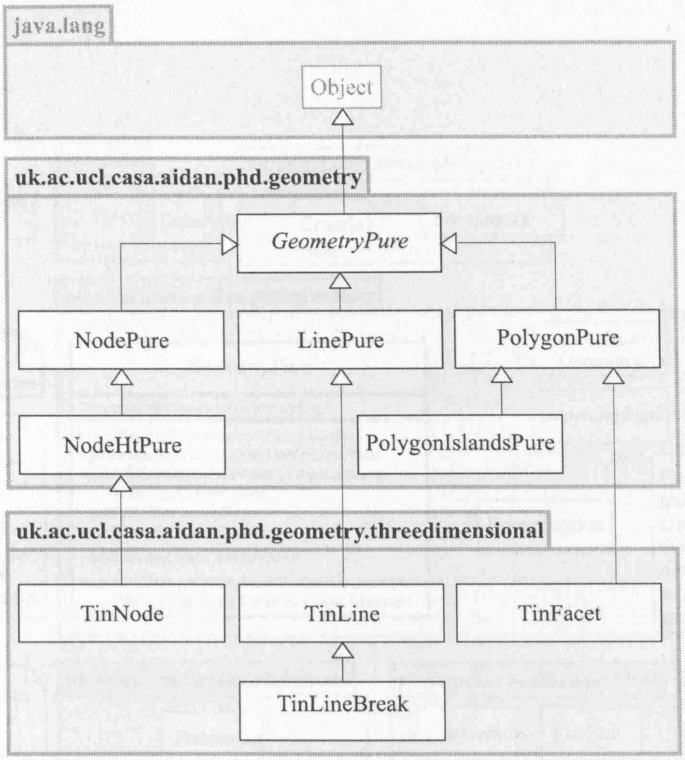


Figure 6.22: The GeometryPure class hierarchy.

6.10.5.1 'GeometryPure'

Figure 6.23 shows the class diagram of the abstract class 'GeometryPure'. As an abstract class, it cannot be instantiated itself, but it contains fields and methods which are inherited by its subclasses. The diagram also shows that 'GeometryPure' extends 'Object'; thus it is a standard Java object rather than a database object. It implements the 'Geometry' interface.

All 'GeometryImpl' subclass instances contain a reference to an instance of 'GeometryPure' (section 6.4.2). The'wrappingGeometry' field stores a reference to the enclosing object (if it exists – it remains empty if not). This is required to pass the methods which are outside the scope of the 'GeometryPure' class hierarchy into the 'GeometryImpl' class hierarchy (figure 6.24). Methods outside the scope of this class hierarchy are 'getFeatures', 'addFeatures' and 'accessThrough'; these operations are specific to the 'GeometryImpl' class hierarchy (the database objects).

The 'getAdj' method returns the immediately adjacent geometries according to a 'Criteria' (section 6.13). To return a list of the adjacent polygons, one can use the query "([is poly])".

The 'getAdjRecurs' method recursively collects and returns a list of adjacent geometries over a wide area according to the parameters. That shown in the UML diagram takes a 'Criteria' and a 'Pedestrian' as parameters and returns all the geometries which conform the 'Criteria' and access constraints.

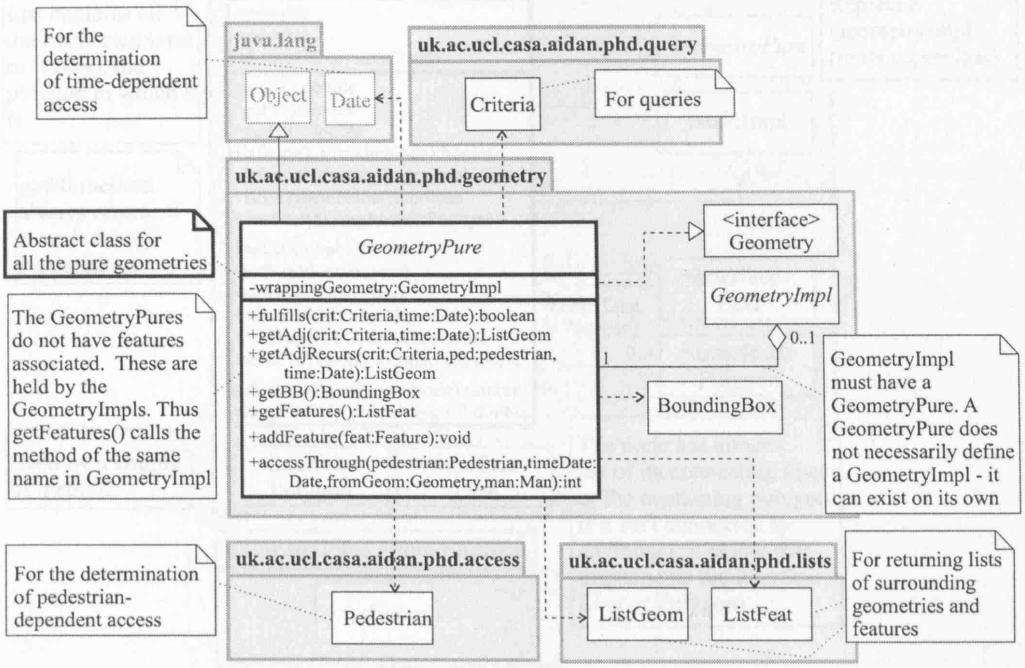


Figure 6.23: UML class diagram of 'GeometryPure', showing its fields, methods and dependencies (note that it is also dependent on the dependencies of its dependencies).

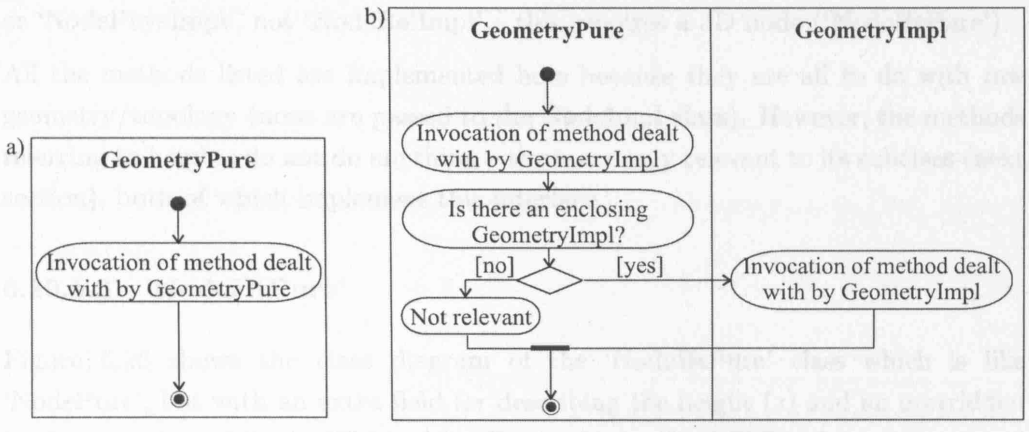


Figure 6.24: UML sequence diagram showing the invocation of a methods dealt with the 'GeometryPure' class hierarchy and the 'GeometryImpl' class hierarchy.

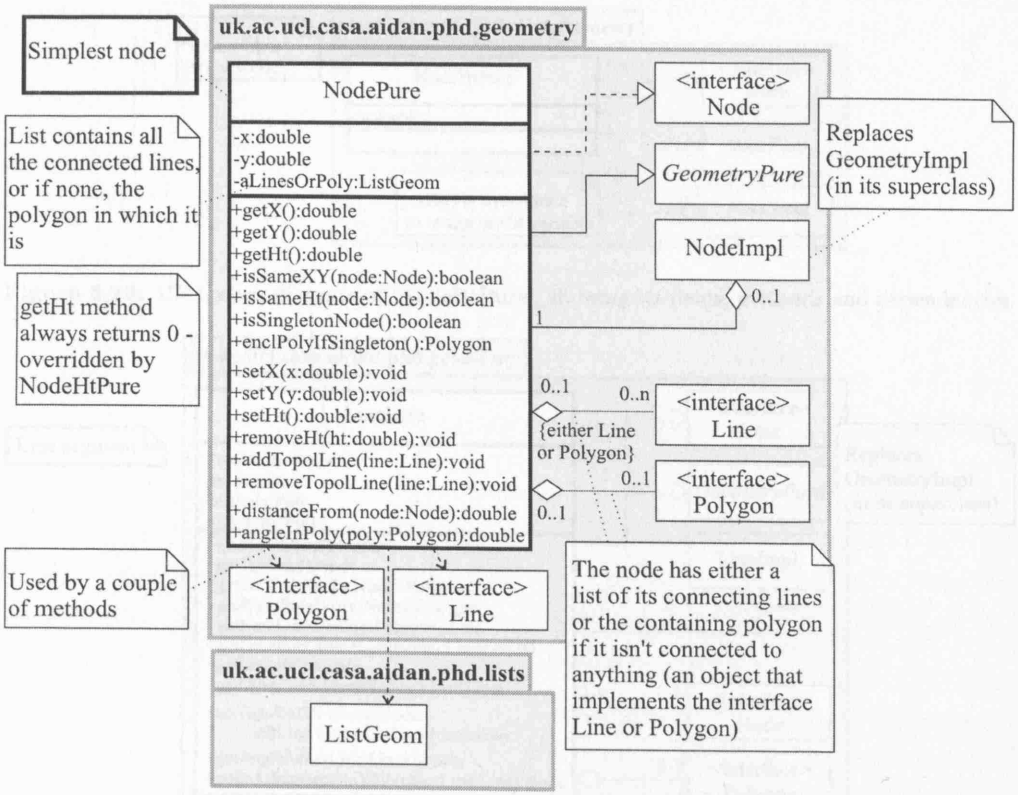


Figure 6.25: UML class diagram of 'NodePure', showing its fields, methods and dependencies.

6.10.5.2 'NodePure'

Figure 6.25 shows the class diagram of the 'NodePure' class. This is the most basic 2D node with fields for its x and y coordinate. Its stored topology is in the form of a list of the connecting lines, or the enclosing polygon if there are none. If it is part of an object from the 'GeomImpl' class hierarchy, the class will be 'NodeImpl' or 'NodePhysImpl', not 'NodeRelImpl' – this requires a 3D node ('NodeHtPure').

All the methods listed are implemented here because they are all to do with raw geometry/topology (none are passed to the NodeImpl class). However, the methods referring to heights do not do anything – these are only relevant to its subclass (next section), both of which implement this interface.

6.10.5.3 'NodeHtPure'

Figure 6.26 shows the class diagram of the 'NodeHtPure' class which is like 'NodePure', but with an extra field for describing the height (z) and an overridden implementation of the 'getHt' and 'setHt' methods. 'NodeHtPure' can be enclosed by 'NodeImpl' and any of its subclasses ('NodeImpl' and 'NodePhysImpl' may enclose either 'NodePure' or 'NodeHtPure').

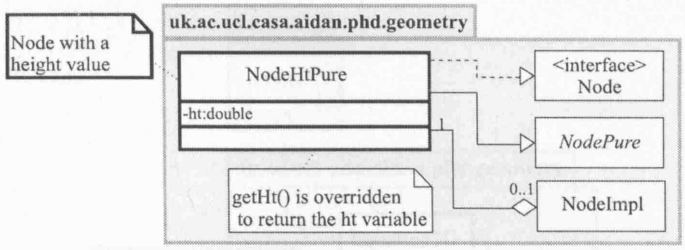


Figure 6.26: UML class diagram of 'NodeHtPure', showing its fields, methods and dependencies.

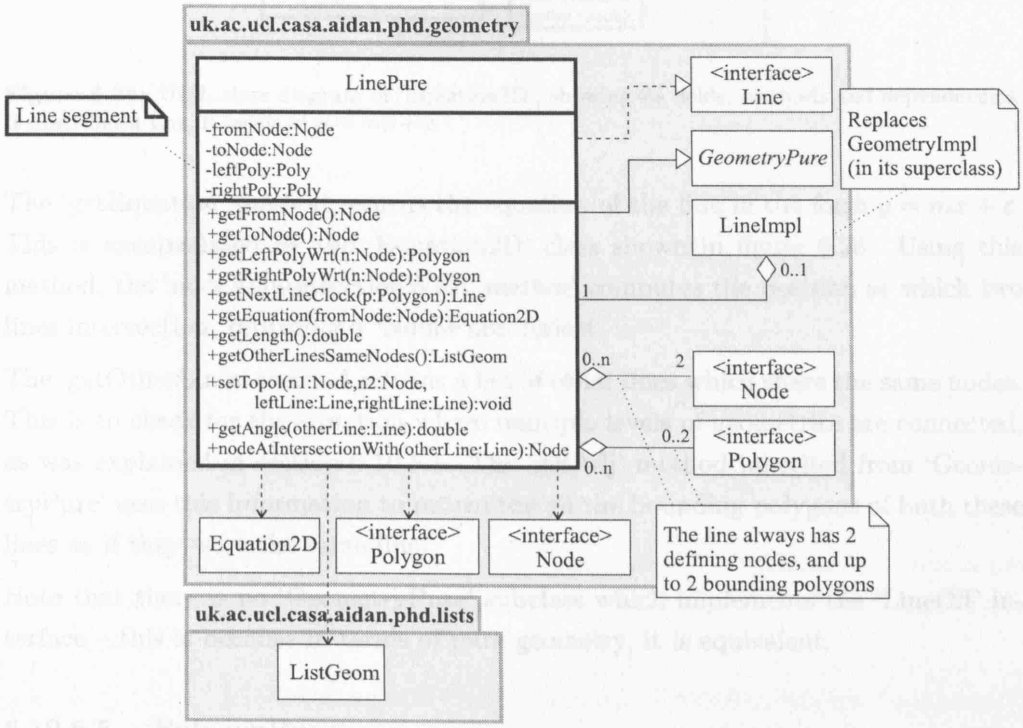


Figure 6.27: UML class diagram of 'LinePure', showing its fields, methods and dependencies.

6.10.5.4 'LinePure'

Figure 6.27 shows the class diagram of the 'LinePure' class. It has four fields describing its topology; the two nodes to which it is attached and the two bounding polygons. If the line lies on the boundary between two polygons, its two adjacent polygon values will be different. If the line is within a polygon rather than on the boundary, its two adjacent polygons will have the same value. If the line is at the edge of the topologically-connected coverage, one of its adjacent polygons will have a value of 'null'.

All the methods which are to do with the raw geometry/topology are implemented in this class. The 'getNextLineClock' method is the key method used to compute the geometry of polygons; it returns the next line in the clockwise order with respect to one of the polygons bounding the line.

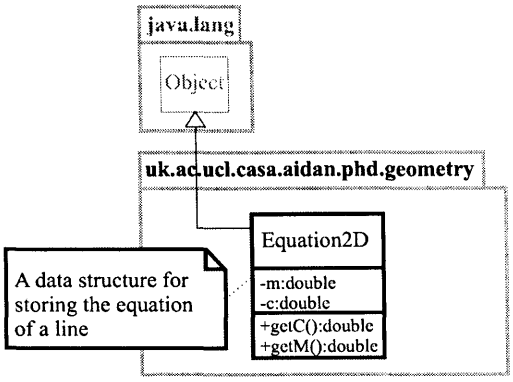


Figure 6.28: UML class diagram of 'Equation2D', showing its fields, methods and dependencies. It describes a line in terms of $y = mx + c$.

The 'getEquation' method returns the equation of the line in the form $y = mx + c$. This is encapsulated by the 'Equation2D' class shown in figure 6.28. Using this method, the 'nodeAtIntersectionWith' method computes the position at which two lines intersection, returning a 'NodePure' object.

The 'getOtherLines' method returns a list of other lines which share the same nodes. This is to check for the situation where multiple levels of geometries are connected, as was explained in section 6.10.5.1. The 'getAdj' method inherited from 'GeometryPure' uses this information to return the all the bounding polygons of both these lines as if they were the same line.

Note that there is no 'GeometryPure' subclass which implements the 'LineOff' interface – this is because in terms of pure geometry, it is equivalent.

6.10.5.5 'PolygonPure'

Figure 6.29 shows the class diagram of the 'PolygonPure' class, the simplest polygon that does not allow islands.

It only has one field, which stores one of the line segments on its boundary (this was mentioned in section 6.4.3.1). This is because there is enough topological information held by the surrounding geometries to derive the geometry of the polygon, and storing all the lines would increase the storage space required. This is to conform to the 'data repository' design principle (section 3.4) that the storage overhead should be low, with derivable data being derived when needed; hence the need for caching polygons (section 6.4.3.1). The method 'getExternalChain' returns a 'Chain' (ordered sequence of line segments; section 6.10.7.3) representing the geometry of the polygon. This is what is cached; full details are in section 6.15.

As is the case for 'NodePure' and 'NodeHtPure', 'PolygonPure' and 'PolygonIslandsPure' (which allows the presence of islands – next section) are different complexities of the same geometrical element (the majority of polygons do not have islands, so this saves storage space), but this does not affect how 'PolygonImpl' works; it may

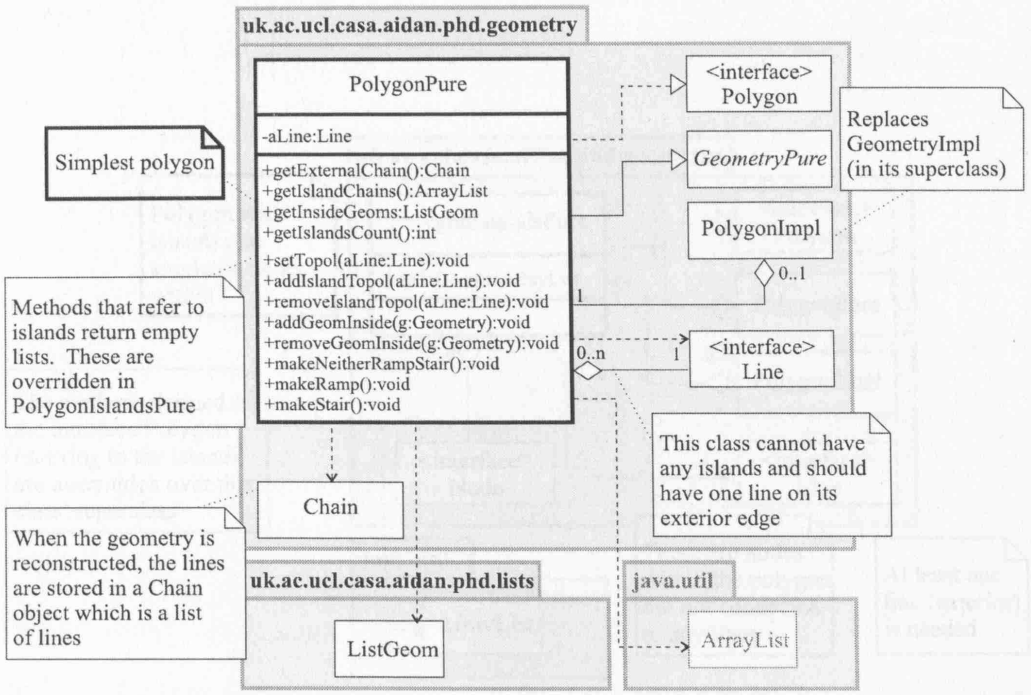


Figure 6.29: UML class diagram of 'PolygonPure', showing its fields, methods and dependencies.

enclose either. This also means that 'PolygonPure' and 'PolygonIslandsPure' share the same interface which means that the simpler one of the two has to implement irrelevant methods (to do nothing). The methods referring to 'inside' geometries are referring to islands – these are relevant for 'PolygonIslandsPure'; they do nothing in 'PolygonImpl'.

The final three methods ('makeNeitherRampStair', 'makeRamp' and 'makeStair') in this class are not relevant to the GeometryPure class hierarchy – their refer to non-geometrical property and are passed through to the enclosing 'PolygonImpl' (section 6.10.6.8) object if present, otherwise they do nothing.

6.10.5.6 'PolygonIslandsPure'

Figure 6.30 shows the class diagram of the 'PolygonIslandsPure' class, representing polygons with islands. The 'insideGeoms' attributes holds a line on the boundary of every island and any 'singleton nodes' (node with no connecting lines). This information is used to delineate the chains for the islands. All the methods related to the inside geometries and islands are implemented in this class (overridden from the superclass).

6.10.6 'GeometryImpl' and its subclasses

The 'GeometryImpl' class hierarchy (figure 6.31) are classes whose instances are directly stored in the database. They do not deal with the raw geometrical/topological

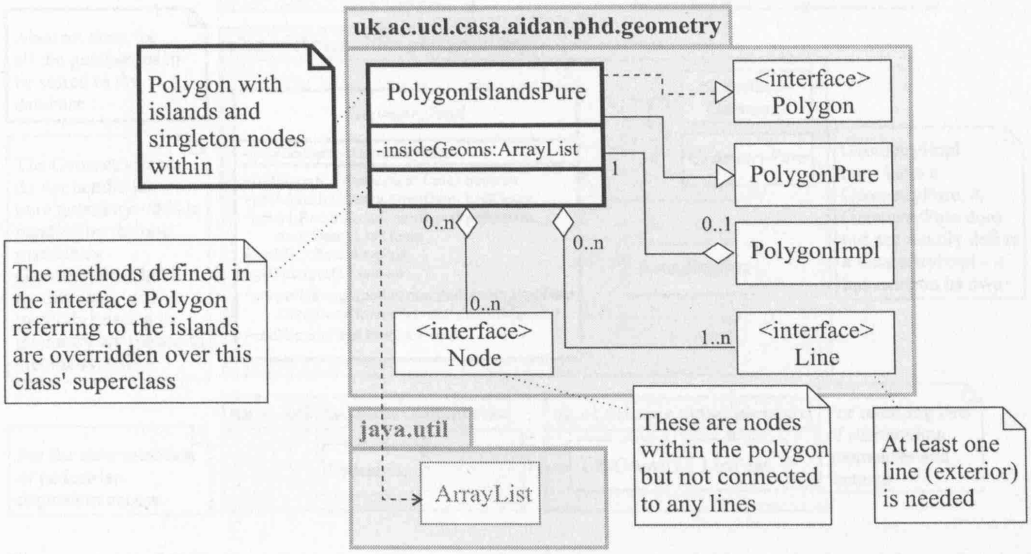


Figure 6.30: UML class diagram of 'PolygonIslandsPure', showing its fields, methods and dependencies.

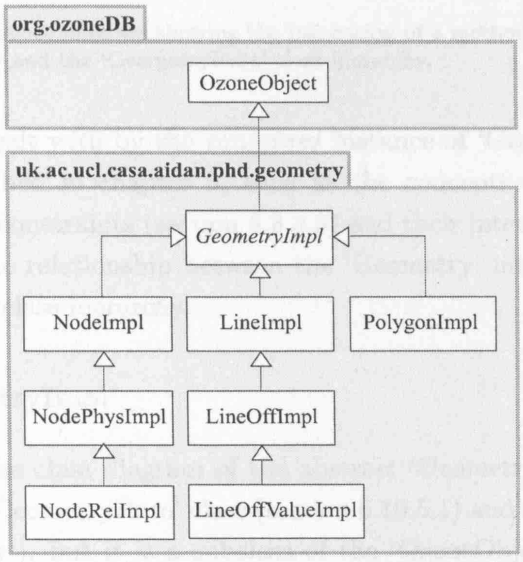


Figure 6.31: The GeometryImpl class hierarchy.

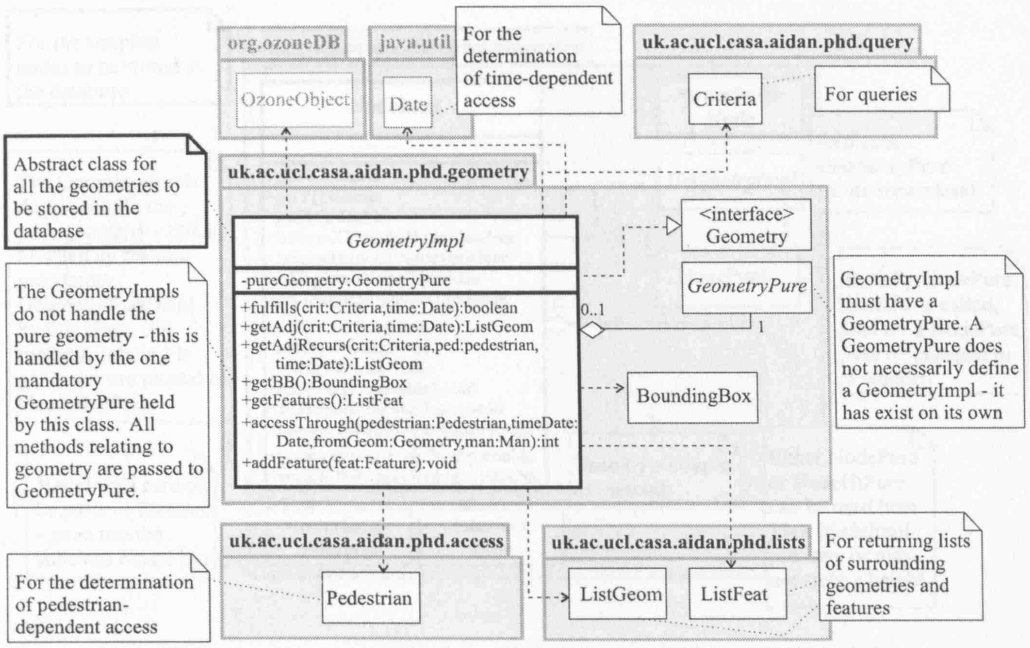


Figure 6.32: UML class diagram of 'GeometryImpl', showing its fields, methods and dependencies.

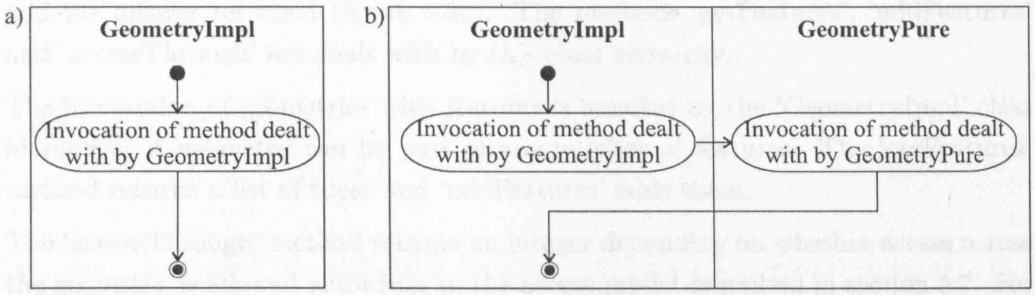


Figure 6.33: UML sequence diagram showing the invocation of a methods dealt with the 'GeometryImpl' class hierarchy and the 'GeometryPure' class hierarchy.

properties (this is dealt with by the *contained* instance of 'GeometryPure'); rather other aspects described in chapter 5, such as the concepts of offset edges (section 5.3.4.2), height constraints (section 5.3.4.3) and their interaction with features. There is a one-to-one relationship between the 'Geometry' interface hierarchy and the 'GeometryImpl' class hierarchy.

6.10.6.1 'GeometryImpl'

Figure 6.32 shows the class diagram of the abstract 'GeometryImpl' class. This is very similar to the 'GeometryPure' class (section 6.10.5.1) and implements the same interface ('Geometry'), but it is a subclass of the 'OzoneObject' class (i.e. it is a database class) and contains a *mandatory* 'GeometryPure' instance.

As discussed in (section 6.10.5.1), some of these methods are dealt with by the mandatory enclosed 'GeometryPure' and these methods are simply passed through

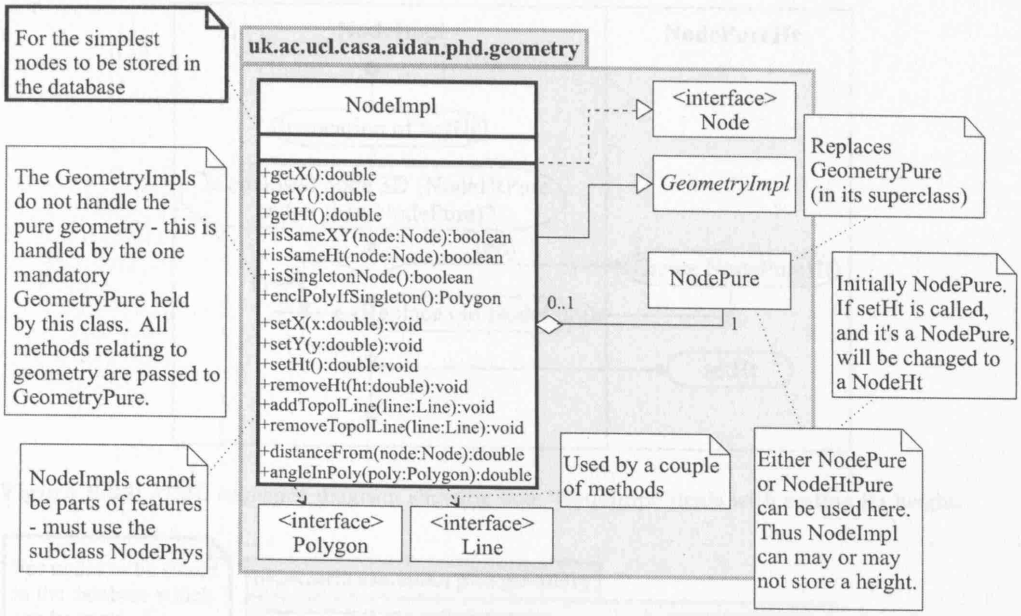


Figure 6.34: UML class diagram of 'NodeImpl', showing its fields, methods and dependencies.

and the answer returned (figure 6.33). The methods 'getFeatures', 'addFeatures' and 'accessThrough' are dealt with by this class hierarchy.

The interaction of geometries with *features* is handled by the 'GeometryImpl' class hierarchy. A geometry can be part of any number of features. The 'getFeatures' method returns a list of these and 'addFeatures' adds them.

The 'accessThrough' method returns an integer depending on whether access across the geometry is allowed according to the access model described in section 5.7. For now, if is sufficient to know that '0' means that full access is allowed and as the number becomes higher, access becomes more difficult. This is partly dependent on the features whose extent the geometry forms part of. Access is discussed further in section 6.14.

6.10.6.2 'NodeImpl'

Figure 6.34 shows the class diagram of the 'NodeImpl' class. Most of the methods are to do with the raw geometry/topology. A NodeImpl may be a 2D or a 3D node, depending on whether it encloses a 'NodePure' or a 'NodeHtPure' instance. By default, a 2D node is enclosed and this is replaced by a 'NodeHtPure' if the 'setHt' node is invoked. The 'setHt' method will then be called on the 'NodeHtPure'. This is illustrated in figure 6.35.

Unique of all the 'GeometeryImpl' subclasses, 'NodeImpl' cannot be part of any features. The reason for this is that the majority of nodes do not belong to features. The 'NodeImpl' type saves storage space by not needing to maintain a list of features. 'NodePhysImpl' is for nodes with features.

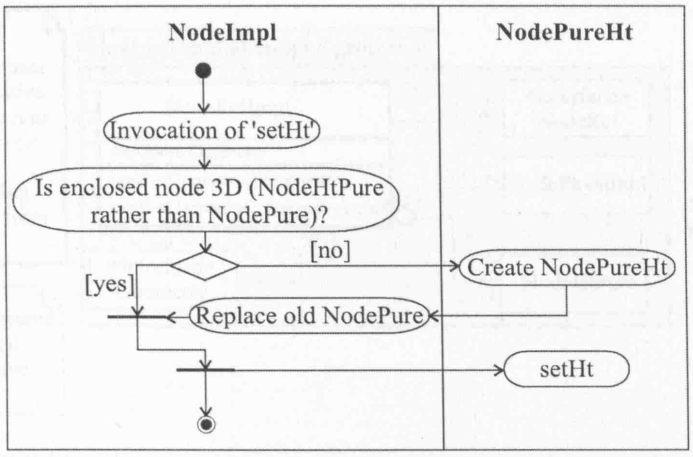


Figure 6.35: UML sequence diagram showing how 'NodeImpl' deals with setting its height.

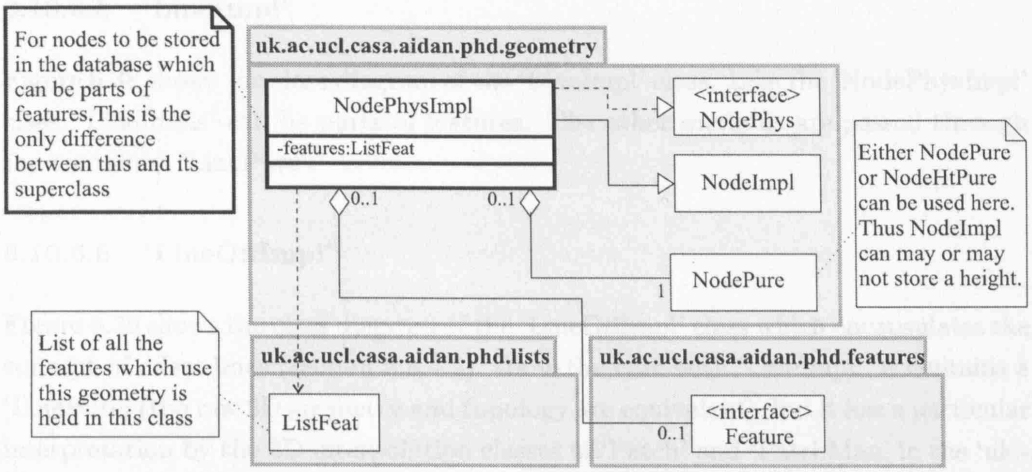


Figure 6.36: UML class diagram of 'NodePhysImpl', showing its fields, methods and dependencies.

6.10.6.3 'NodePhysImpl'

Figure 6.36 shows the class diagram of the 'NodePhysImpl' class. The only difference from 'NodeImpl' is its ability to be parts of features, through the 'features' field.

6.10.6.4 'NodeRelImpl'

Figure 6.37 shows the class diagram of the 'NodeRelImpl' class. This class must always contain a 'NodeHtPure', but the height is interpreted as a *relative height* with respect to the height of 'refGeom' field. Relative heights are an important concept in the conceptual model, as explained in section 5.3.4.3. This class takes the 3D coordinates defined by 'NodeHtPure' and adds a different interpretation on the height, making it a *relative* rather than *absolute* height. This interpretation of height is used by the 3D interpolation classes of 'Patch' and 'PatchMan' in the 'uk.ac.ucl.casa.aidan.geometry.threedimensional' package (section 6.11).

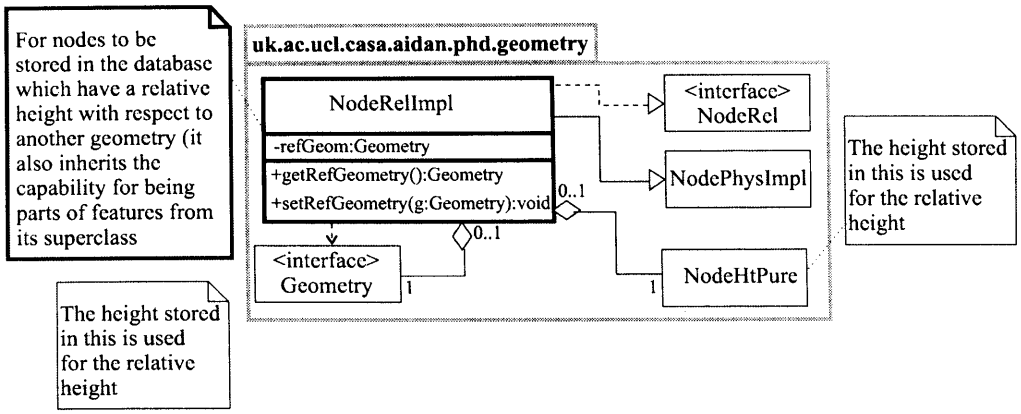


Figure 6.37: UML class diagram of 'NodeRelImpl', showing its fields, methods and dependencies.

6.10.6.5 'LineImpl'

Figure 6.38 shows the class diagram of the 'LineImpl' class. Like the 'NodePhysImpl' class, 'LineImpls' can be parts of features. The other methods are passed through the contained 'LinePure'.

6.10.6.6 'LineOffImpl'

Figure 6.39 shows the class diagram of the 'LineOffImpl' class which encapsulates the concept of offset lines (section 5.3.4.2). As is the case with 'LineImpl', it contains a 'LinePure' (the raw 2D geometry and topology are equivalent), but it has a particular interpretation by the 3D interpolation classes of 'Patch' and 'PatchMan' in the 'uk.ac.ucl.casa.aidan.geometry.threedimensional' package (section 6.11).

6.10.6.7 'LineOffValueImpl'

Figure 6.40 shows the class diagram of the 'LineOffValueImpl' class. It functions as the 'LineOff' class, its only difference being that the offset height (a relative height) is specified and fixed along the line, relative to one of its adjacent polygons which is specified geometry with the 'setRefGeometry' method. As noted, this class is not strictly necessary (it is not in the conceptual mode), because the same affect can be obtained by using a 'LineOffImpl' instance with 'NodeRelImpl' instances at its start and finish. It is provided by pragmatic convenience.

Shared lines (where two input layers join) are treated as breaklines ('LineOffValues' with a height value of zero) by 'Patch' and 'PatchMan' (the 3D geometry resolving routines).

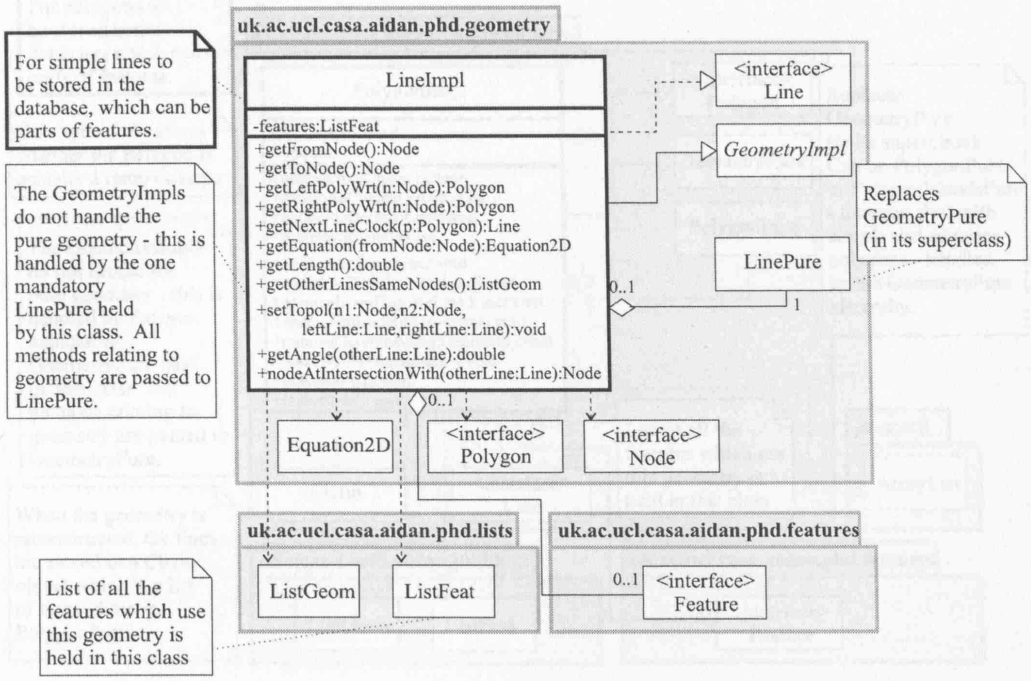


Figure 6.38: UML class diagram of 'LineImpl', showing its fields, methods and dependencies.

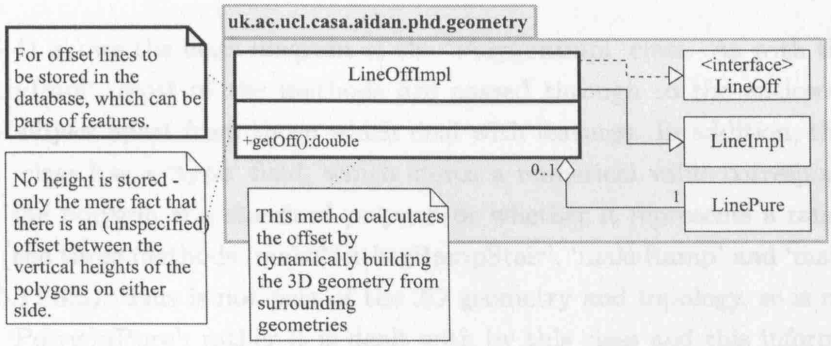


Figure 6.39: UML class diagram of 'LineOffImpl', showing its fields, methods and dependencies.

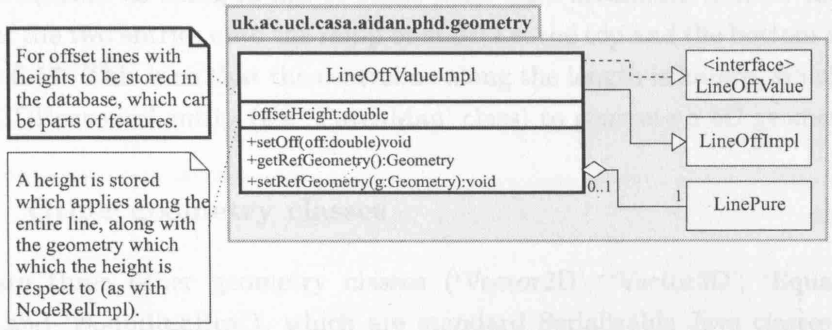


Figure 6.40: UML class diagram of 'LineOffValueImpl', showing its fields, methods and dependencies.

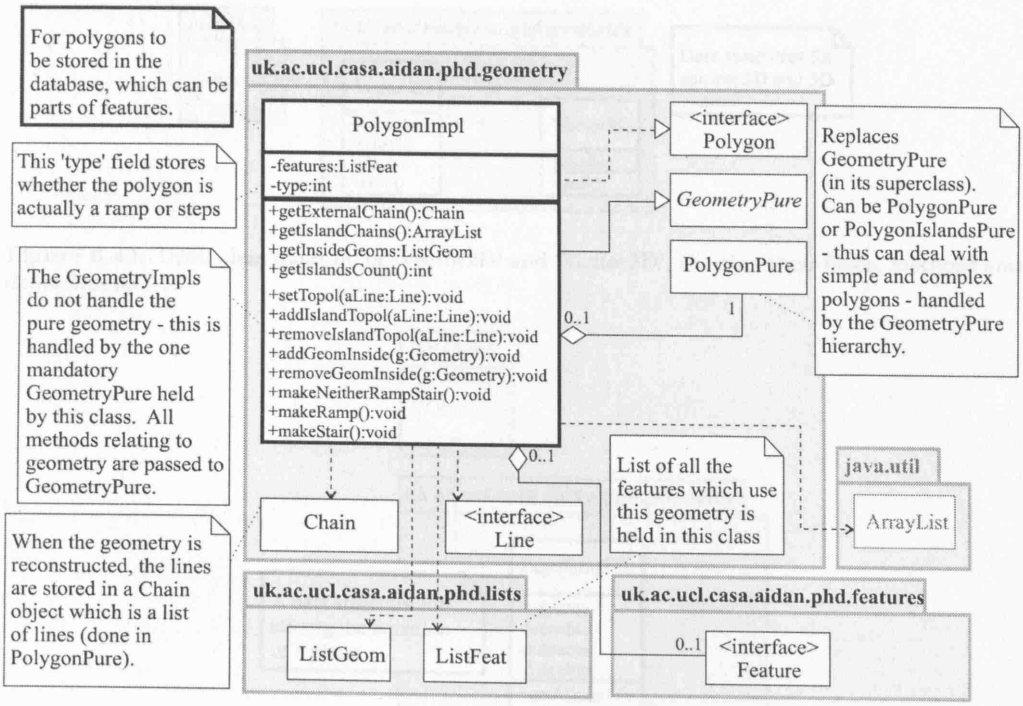


Figure 6.41: UML class diagram of 'PolygonImpl', showing its fields, methods and dependencies.

6.10.6.8 'PolygonImpl'

Figure 6.41 shows the class diagram of the 'PolygonImpl' class. As with the other 'GeometryImpl', most of the methods are passed through to the enclosed 'PolygonPure' object apart from those which deal with features. In addition, the 'PolygonImpl' class has a 'type' field, which stores a numerical value corresponding to whether the polygon is a standard polygon or whether it represents a ramp or set of steps (the three methods 'makeNeitherRampStair', 'makeRamp' and 'makeStair'; section 6.11.3.3). This is not part of the 2D geometry and topology, so is not dealt with by 'PolygonPure'; rather it is dealt with by this class and this information is used by the 3D interpolation classes of 'Patch' and 'PatchMan' in the 'uk.ac.ucl.casa.aidan.geometry.threedimensional' package (section 6.11).

Polygons marked as being ramps or stairs must have breaklines to show the whereabouts of the two entries onto the ramp or stair; i.e. the top and the bottom as shown in figure 6.57. This is so that the centreline along the length is known, which is used by the '3DReasoner' entity (the 'PatchMan' class) to generate a 3D geometry.

6.10.7 Other geometry classes

There are three other geometry classes ('Vector2D', 'Vector3D', 'Equation3D', 'Chain' and 'BoundingBox'), which are standard Serializable Java classes. These do not directly correspond to entities in the conceptual model.

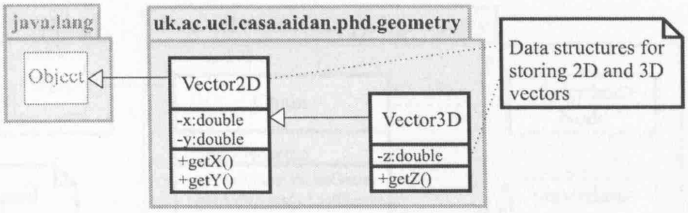


Figure 6.42: UML class diagram of 'Vector2D' and 'Vector3D', showing their fields, methods and dependencies.

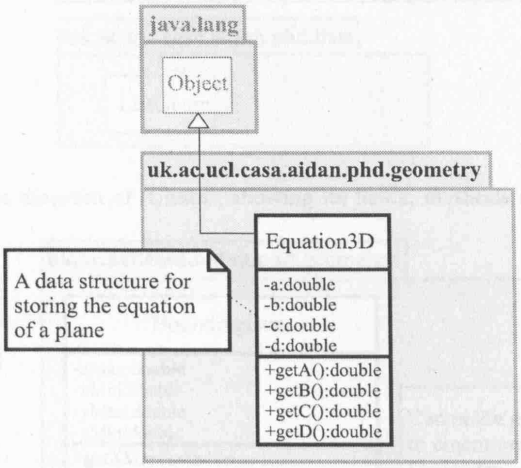


Figure 6.43: UML class diagram of 'Equation3D', showing its fields, methods and dependencies. It describes the equation of a plane in the form $y = ax + by + cz + d$.

6.10.7.1 'Vector2D' and 'Vector3D'

Figure 6.42 shows the class diagrams for 'Vector2D' and 'Vector3D'. These are used by some methods for representing lines in 2D and 3D.

6.10.7.2 'Equation3D'

Figure 6.43 shows the class diagram of the 'Equation3D' class, which describes the equation of a plane in the form $y = ax + by + cz + d$. Otherwise, it is very similar to the 'Equation2D' class (section 6.10.5.4).

6.10.7.3 'Chain'

Figure 6.43 shows the class diagram of the 'Chain' class, an ordered sequence of lines. Unclosed 'Chains' are used to represent polylines and closed 'Chains' are used to represent the outlines of polygons and their islands. The methods 'getAsNodes' and 'getAsLines' allow the 'Chain' so be treated as an ordered list of nodes or an ordered list of lines.

This class is used by the 'getExternalChain' and 'getIslandChains' methods of classes which implement the 'Geometry' interface (section 6.10.1) and cached by 'Poly-

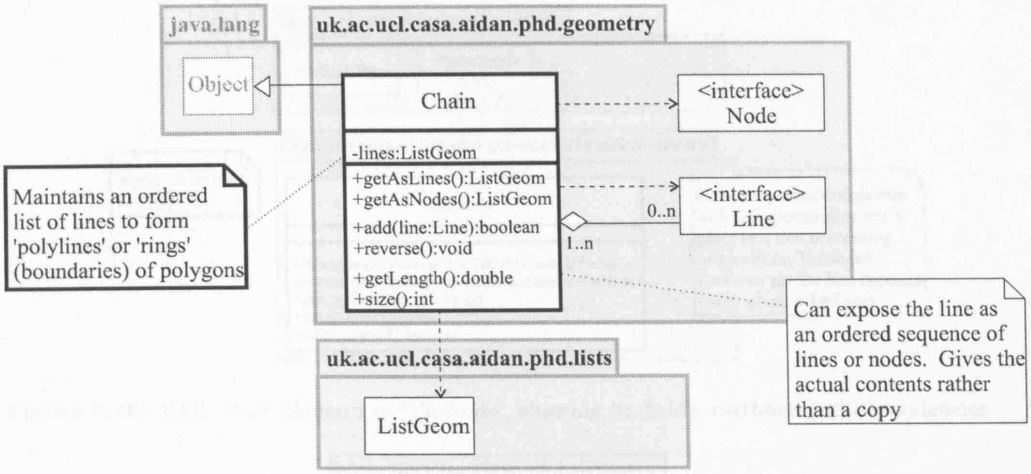


Figure 6.44: UML class diagram of 'Chain', showing its fields, methods and dependencies.

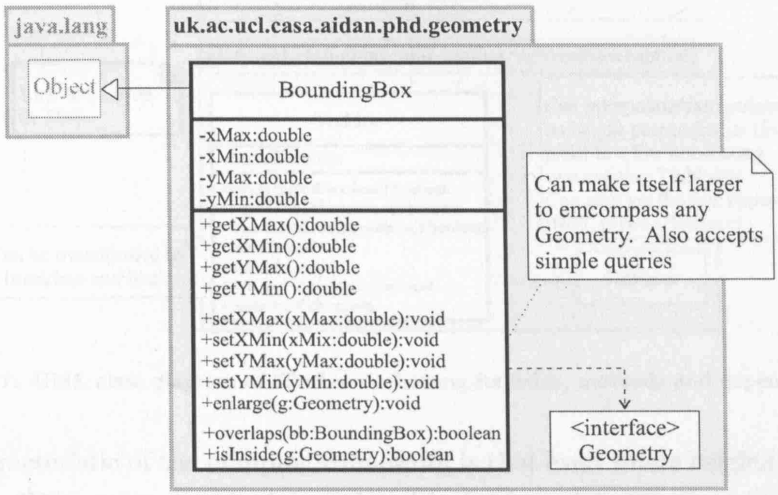


Figure 6.45: UML class diagram of 'BoundingBox', showing its fields, methods and dependencies.

gonCached' (section 6.15.2.2). It is also used by the 'ShapePolygon' class which models polygons for writing to Shapefiles (section 6.7).

6.10.7.4 'BoundingBox'

Figure 6.45 shows the class diagram of the 'BoundingBox' class. This used internally by some methods.

6.11 3D Geometry (uk.ac.ucl.casa.aidan.phd.threedimensional)

The 'PatchMan' class is responsible for coordinating the 3D interpolation and extrapolation of the geometry and it supplies the resulting 3D geometry to the rest of the mapping framework.

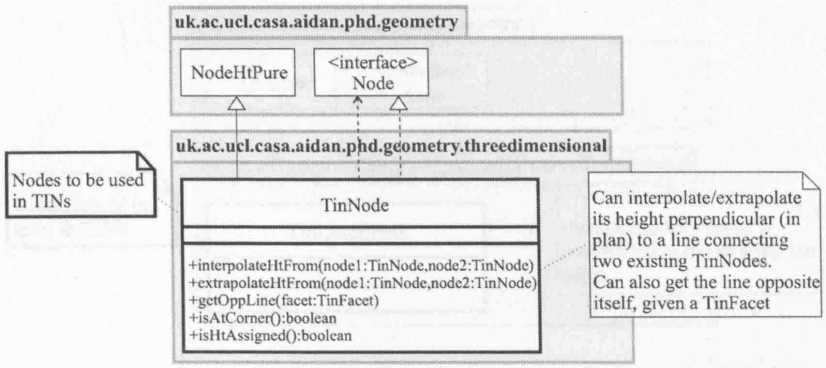


Figure 6.46: UML class diagram of 'TinNode', showing its fields, methods and dependencies.

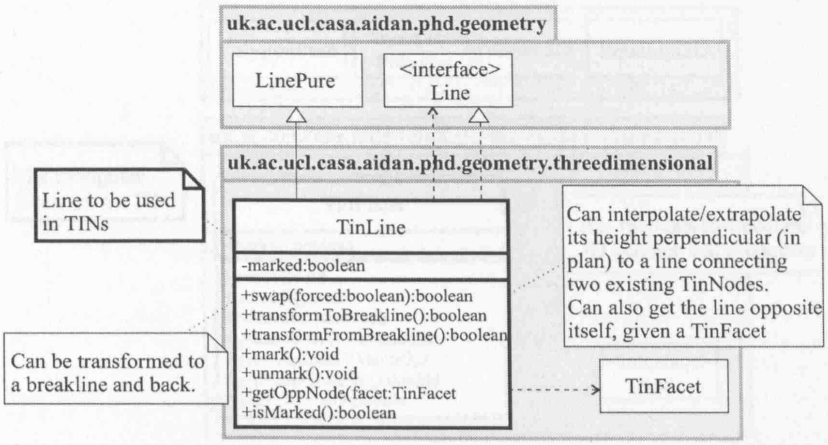


Figure 6.47: UML class diagram of 'TinLine', showing its fields, methods and dependencies.

A key characteristic of the mapping framework is that even where heights are under-resolved, a 3D geometry can be interpolated and extrapolated from a series of height constraints (section 5.3.4). For this reason, the 3D interpolation/extrapolation is computed dynamically only *where and when it is required.*

The task of 'PatchMan' is supported by the other classes in this package. The 'Patch' class encapsulates the *patch* concept (section 5.4.2) and is responsible for building a 3D surface model of its geometry, taking into account the surface geometry of its surrounding patches. The surface geometry itself is encapsulated by the 'Tin' class which represents it as a 2.5D surface. A 'Tin' instance contains a set of TIN elements encapsulated by the classes of 'TinNode', 'TinLine', 'TinLineBreak' and 'TinFacet' (section 6.11.1).

6.11.1 Surface interpolation (TIN elements)

A topologically-structured set of TIN elements comprise a TIN. These elements are instances of 'TinNode', 'TinLine', 'TinLineBreak' and 'TinFacet' and they are part of the 'GeometryPure' class hierarchy as shown at the bottom of figure 6.22; thus they inherit the geometrical and topological properties, adding methods relevant

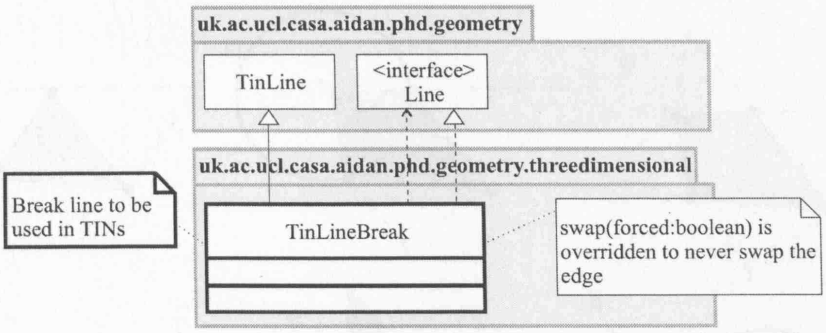


Figure 6.48: UML class diagram of 'TinLineBreak', showing its fields, methods and dependencies.

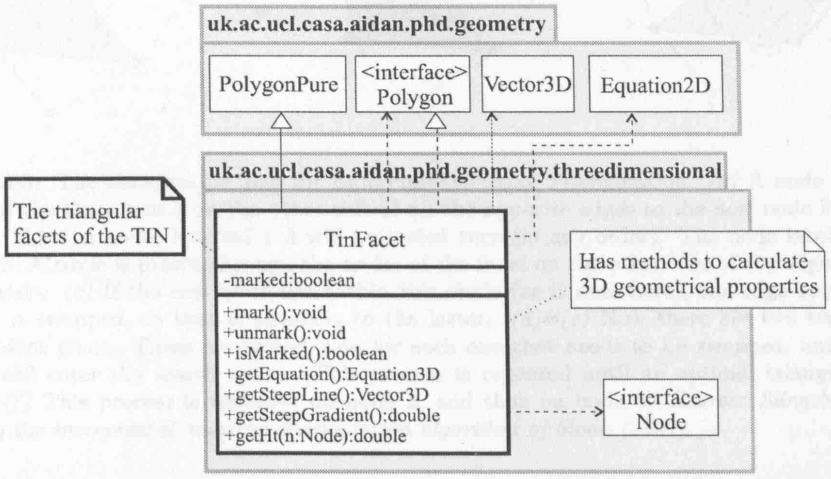


Figure 6.49: UML class diagram of 'TinFacet', showing its fields, methods and dependencies.

to TINs. 'TinNode' (figure 6.46) has additional methods for returning the opposite 'TinNode' within a specified 'TinFacet' and methods used internally to interpolating or extrapolating a height perpendicular to a line connecting two specified nodes. 'TinLine' (figure 6.47) can be 'swapped' (changed to its alternative configuration) according to the algorithm of Sloan (1987), marked (temporarily flagged; this is used internally), can return the 'TinNode' opposite it in a specified 'TinFacet' and can transform itself into a breakline. 'TinLineBreak' (figure 6.48)is similar to 'TinLine' except that its configuration cannot be changed so the 'swap' method is overridden to prevent the configuration from changing. 'TinFacet' (figure 6.49), which can be oriented in 3D can be marked, can calculate its gradient and aspect and can interpolate a height from its surface from the 2D position of a node passed to it.

6.11.2 Surface interpolation ('Tin')

As described in section 5.3.3, the conceptual model models space as a multilayered set of 2.5D surfaces which are topologically joined at various places (illustrated in figure 5.6). Section 4.5.2.1 describes how triangular irregular networks (TINs) can be used to represent 2.5D surfaces. The advantage of using a TIN over a raster is that the original data points can be preserved and the TIN can be built incrementally.

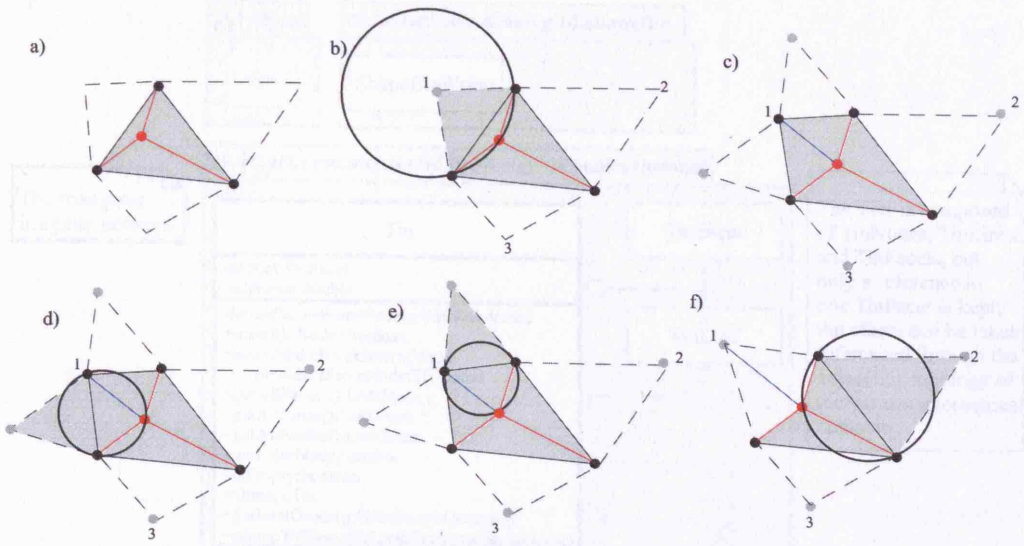


Figure 6.50: The circumcircle Test for Enforcing Delaunay Triangulation. *(a)* A node has been added, and the three facets on the other side of all the opposite edges to the new node have been identified. *(b)* The nodes labelled 1–3 will be tested turn (in any order). The node labelled '1' is tested first. A circle is drawn through the nodes of the facet on the other side of the edge opposite the new node. *(c)* If the new node lies within this circle (as it does here), the edge opposite the new node is swapped, so that is connects to the latter. *(d)&(e)* Now there are two triangles in the equivalent place. These are tested, and for each one that needs to be swapped, another two triangles will enter the search space. This process is repeated until an optimal triangulation is reached. *(f)* This process is repeated on node 2, and then on node 3. *Source: Slingsby (2002), describing the incremental addition triangulation algorithm of Sloan (1987).*

The 'Tin' class encapsulates the concept of a TIN, which can build, maintain and provide information about itself. Every patch's (section 5.4.2) height is described as a continuous surface using an instance of 'Tin'.

Surfaces in TINs are represented by triangulating a set of spot heights and (usually) linearly interpolating the height across the surface. Sets of points can be triangulated in different ways; the most common criterion for doing this is to join the nearest points together, such that the resulting triangles are as 'fat' as possible making the interpolation as local as possible. The Delaunay Triangulation produces a triangulation with these properties; more formally, a circle passing through each point of a triangle must not encompass any other point. Figure 6.50 illustrates how a Delaunay Triangulation can be built, using an algorithm described by Sloan (1987). As well as the advantage that height interpolation is as local as possible another advantage is that Delaunay Triangulations are unique (except in some special cases).

TINs can be built from sets of points. The points used may come from original ground surveys, digitised from contours or taken from remote sensed imagery. Lee (1991) reviews methods for selecting significant points from remote sensed imagery.

In Delaunay Triangulations, the triangulation and thus the position of triangles edges is purely dependent on the distribution of points. The Delaunay criterion to produces a good global surface but does not take into account linear morphological surface structures. Breaks of slope are linear morphological structures which are of great

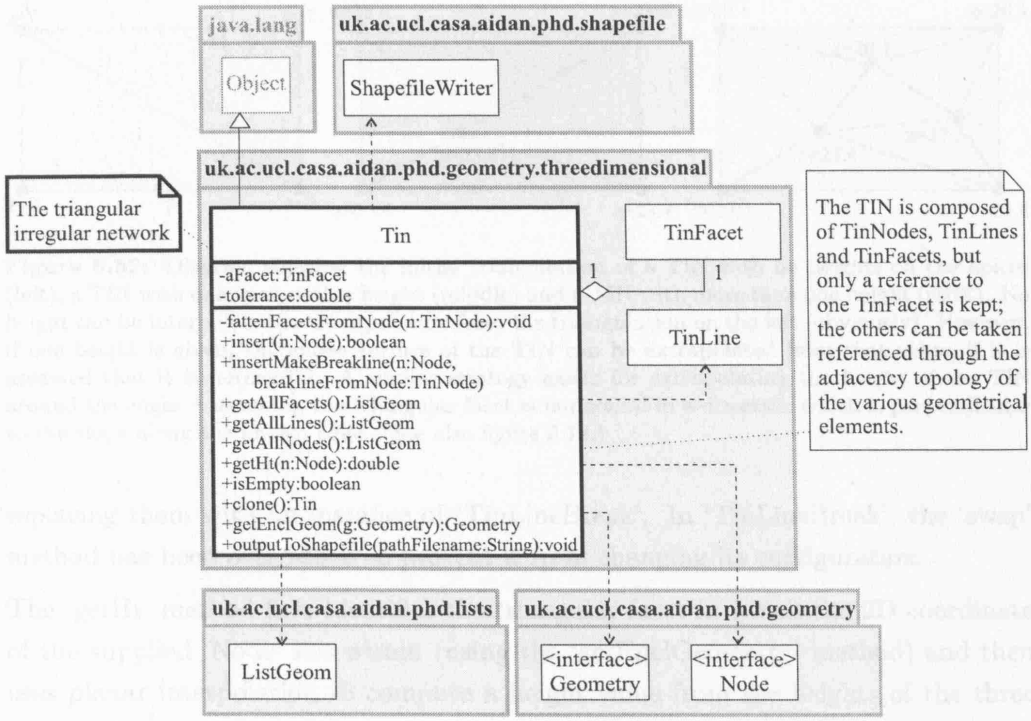


Figure 6.51: UML class diagram of 'Tin', showing its fields, methods and dependencies.

importance to surface geometry. Figure 4.6 shows the importance of a constrained triangulation for studying drainage patterns on TIN-based terrain models and it is clear that this importance stretches to large-scale terrain modelling. Constrained TINs preserve linear surface structures by constraining particular triangle edges so that they coincide with linear surface structures. This results in a triangulation where the Delaunay criteria is generally applied, but relaxed around constrained triangle edges.

Figure 6.51 shows the class diagram of the 'Tin' class, which comprises a topologically-connected set of instances of 'TinNode', 'TinLine', 'TinLineBreak' (a breakline) and 'TinFacet' (triangle over which the height is interpolated from its apices), all of which are part of the 'GeometryPure' hierarchy. As such they inherit the geometrical and topological properties of the geometries and are not database objects. An instance of 'Tin' itself only holds a reference to one 'TinFacet'; the rest can be retrieved using the topological information. The algorithm of Sloan (1987) is used to build the TIN in a similar implementation to that used by Slingsby (2002). When an instance of 'TIN' is created, there is an initial triangulation of two triangles which form a rectangle encompassing the area in which eventual surface will be. The vertices are not assigned heights, a situation illustrated on the left in figure 6.52. The method 'insert' takes a node with a height and inserts it into the TIN, automatically updating the Delaunay Triangulation as shown in figure 6.52. The Delaunay Triangulation is updated using the 'swap' method of 'TinLine' and the circumcircle test developed by Sloan (1987). Triangle edges ('TinLines') can be constrained by

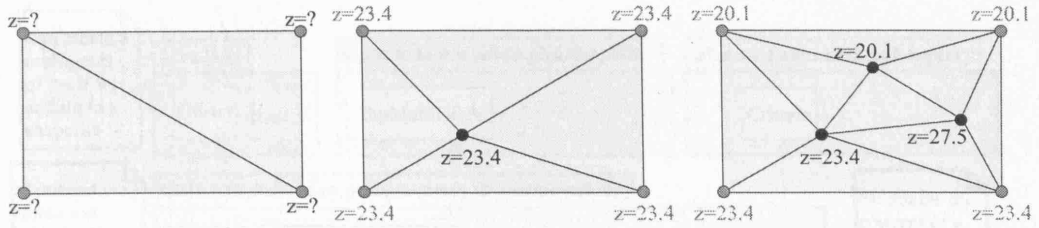


Figure 6.52: Diagram showing the initial triangulation of a TIN with no heights on the apices (left), a TIN with one apex with a height (middle) and a TIN with more than one height (right). No height can be interpolated or extrapolation from the triangulation on the left (obviously). However, if one height is given, the entire surface of the TIN can be extrapolated from that value, if it is assumed that it is horizontal. A similar strategy exists for extrapolating the height of the TIN around the edges – assuming the triangular facet is horizontal in a direction which is perpendicular to the slope along the known edge. (See also figure 5.13.)

replacing them with an instance of 'TinLineBreak'. In 'TinLineBreak', the 'swap' method has been overridden to prevent it from changing its configuration.

The 'getHt' method first identifies the triangular facet in which the 2D coordinate of the supplied 'Node' lies within (using the 'getEnclGeometry' method) and then uses planar interpolation to compute a height value from the heights of the three apices of the triangular facet. In order support the requirements of the mapping framework, an unusual modification has been made to the TIN, which is illustrated in figure 6.52. The initial triangulation encompassing the whole area to which the TIN applies uses 'TinNodes' with no heights assigned. When one or more 'TinNodes' (with heights) are added, any facet (around the outside of the TIN) with a node whose height is not assigned is assigned a height based on the assumption that the facet is horizontal. The reasons for this are described in section 5.4.1 and illustrated in figure 5.13.

6.11.3 'Patch'

As explained, the 'PatchMan' class is responsible for coordinating the 3D interpolation and extrapolation of the geometry of the mapping framework. 'Patch' is a *private inner class* (defined within 'PatchMan'), thus is inaccessible from outside 'PatchMan'. Its class diagram is shown in figure 6.53.

A patch is a contiguous set of geometries whose surface is continuous and can be described by a 2.5D surface. This is described in section 5.4.2 and illustrated in figure 5.14.

A 'Patch' is capable of writing out a 'ShpMultiPatch' (a 3D data structure for 3D Shapefile output) through the 'getMultiPatch' method. This writes out all the geometry of a patch in 3D according to the TIN.

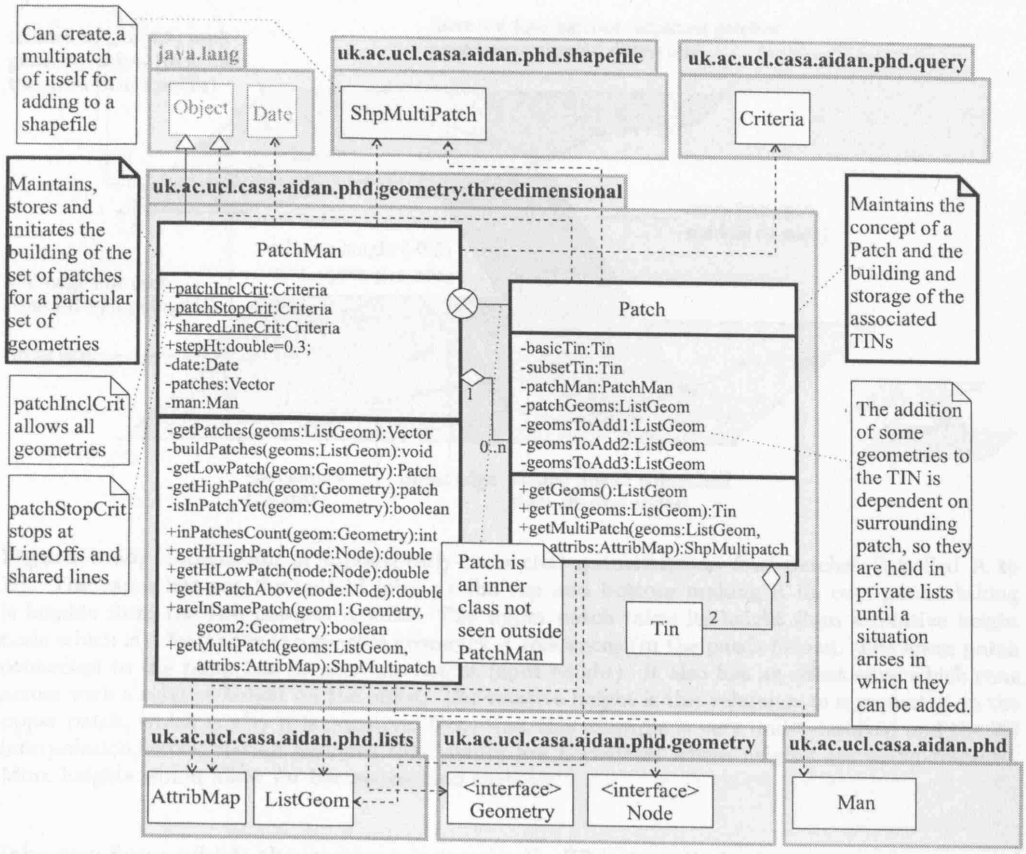


Figure 6.53: UML class diagram of 'PatchMan' and its inner class 'Patch', showing their fields, methods and dependencies.

6.11.3.1 Patch definition

The concept of patches does not exist in the database. Instead they are temporarily created and built in RAM, only to producing a 3D geometry. Patches are automatically defined and delineated from all the topologically-connected geometry; this is shown in figure 6.54. Figure 6.53 includes the class diagram of the 'Patch' class. The definition of a patch is held by the two static variables 'patchInclCrit' and 'patchStopCrit' in 'PatchMan'. 'PatchInclCrit' is set as "([is geom])", which is fulfilled by all geometries. 'patchStopCrit' is set as "([is linenullpoly] or [is lineoff] or [is lineshared])", which is fulfilled by any geometry which is either:

- a line which does not bound two polygons;
- an offset line (section 6.10.6.6); or
- a line with coincident lines sharing the same nodes (section 6.10.5).

There is a version of the 'getAdjRecurs' method defined in the 'Geometry' interface which recursively collects all the adjacent geometries which are fulfilled by one 'Criteria', up to and including those fulfilled by another 'Criteria' (this particular version of 'getAdjRecurs' no listed in this section). It is this version of this method which is used to collect all the geometries in the same patch as the starting geometry

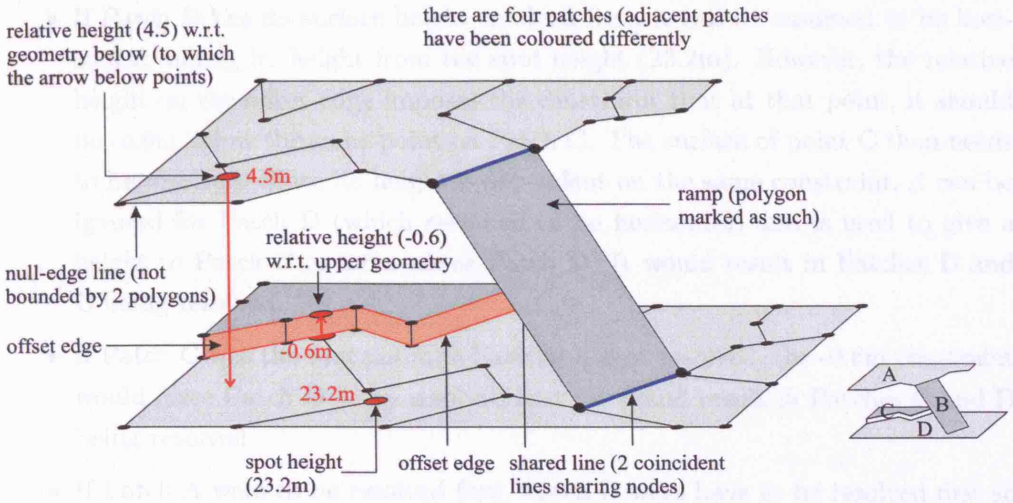


Figure 6.54: Illustration of topologically-connected geometries on four patches (labelled A to D). The ramp has non-2-manifold joins at the top and bottom making it its own patch, taking it heights from the two patches it joins. The upper patch takes its height from a relative height node which is 4.5m above the specific geometry it references (in the patch below). The lower patch connected to the ramp has an absolute height (spot height). It also has an offset edge which runs across with a relative height on the offset. The relative height is this reference to a geometry on the upper patch, which is why it is negative. Note that this example is very under-resolved and the 3D interpolation/extrapolation assumes that planes are horizontal where heights are under-resolved. More heights would allow for better-resolved surfaces.

(the one from which the method is invoked). This is called when a new instance of 'Patch' is created and the resulting list of geometries is used to populate the private field 'patchGeoms'

6.11.3.2 Generating a 2.5D patch surface

Since 3D geometry is only computed *when needed*, no attempt to build TIN is made until required. A call to the 'getTin' method of 'Patch' checks to see whether the TIN has been built (in the instance of 'PatchMan') and returns it if so. If not, the algorithm to build a TIN is initiated and the result stored and returned.

Although 'Patch' is only responsible for computing a surface for itself, its dependence on the surfaces of its surrounding patches makes this quite a complicated problem. Figure 6.54 shows some topologically-connected geometry from which four patches have been identified. It can be seen that:

- Patch A is directly dependent on Patch D
- Patch B is directly dependent on Patch A and D
- Patch C is directly dependent on Patch D
- Patch D *may* be directly dependent Patch C

This can be quite a complicated problem because dependencies can propagate out from the initial patch.

- If Patch D has its surface height resolved first, it can be assumed to be horizontal, taking its height from the spot height (23.2m). However, the relative height on the offset edge imposes the constraint that at that point, it should be -0.6m below the same point on Patch C. The surface of point C then needs to be resolved. Since its height is dependent on the same constraint, it can be ignored for Patch D (which assumed to be horizontal) and is used to give a height to Patch C or 0.6m *above* Patch D. It would result in Patches D and C being resolved.
- If Patch C was the first patch to have its height resolved, the -0.6m constraint would force Patch D to be resolved first. It would result in Patches C and D being resolved.
- If Patch A were to be resolved first, Patch D were have to be resolved first so that the relative height could be resolved. It would result in Patches A, D and C being resolved.
- If Patch B were to be resolved first, it would require Patches A and D to be resolved. It would result in all Patches being resolved.

One can envisage more complicated examples, where there are many more height constraints and a convoluted web of dependencies. Inevitability, there will be cases where it is impossible to calculate a height. This would be the case if the spot height in Patch D was missing.

The order in which the 'NodeHts' and 'NodeOffs' are processed is also important. Clearly, processing heights which are not dependent on other heights should be done before heights which are dependent. Also, relative heights along offset lines which are dependent on a geometry *in* the current patch should have no bearing on the height of the current patch, but should affect the other patch. For example, in figure 6.54, the relative height on the offset edge between Patches C and D references Patch C, so it should have no bearing on Patch C.

However, in the case of the relative height between Patches C and D, Patch C has no other height information. Thus it can only take its height from this relative height. Only under similar circumstances can a relative height which references a geometry in the same patch affect the surface geometry of the patch. This is of course dependent on Patch D having a height.

The solution to these considerations is for a patch to maintain a list of the 'NodeOffs' and 'NodeHts' it has yet to add. These are added in three lists in order of priority of adding. When one of these is successfully added, it is removed from the 'to add' list. The 'getTin' method may recursively be called several times, on other Patches before the height is eventually resolved. As these methods are called on each other, the TINs become more and more complete until they are eventually complete. These TINs are then stored in the instance of 'Patch', which is maintained by the 'PatchMan' class (section 6.11).

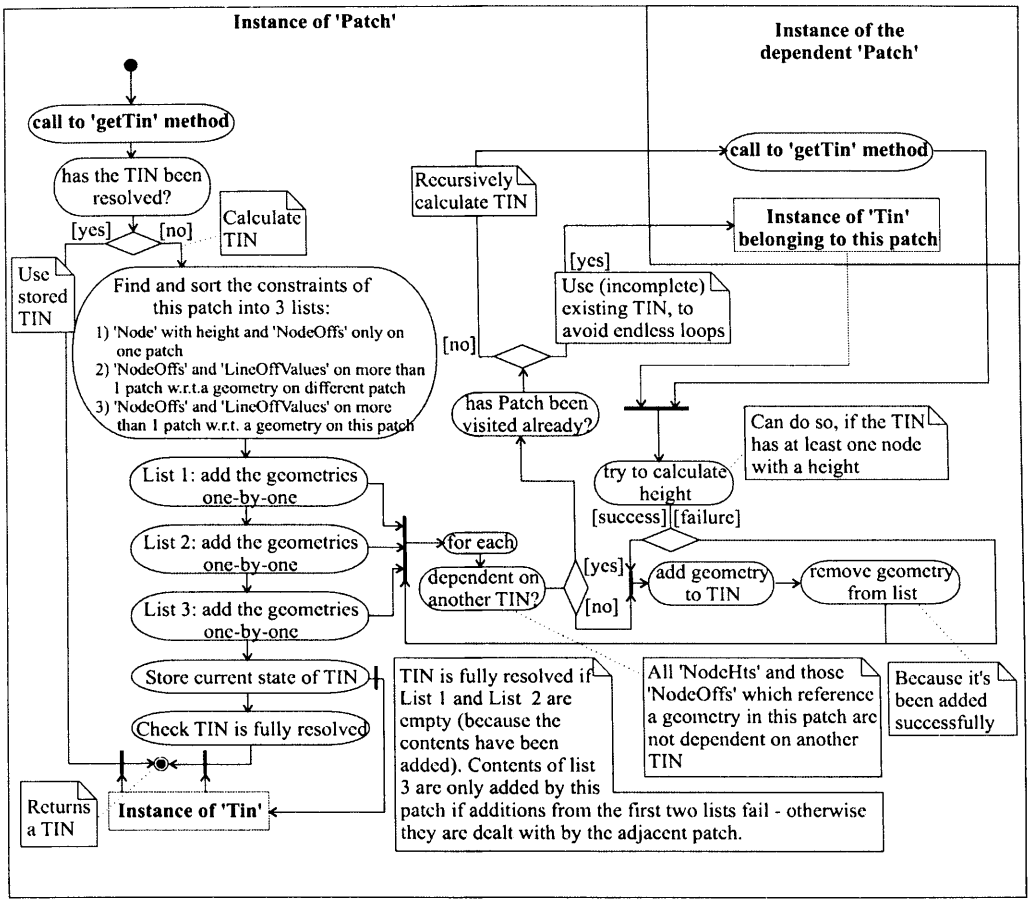


Figure 6.55: UML sequence diagram showing how TINs are built by patches.

The process of building the TIN for patches is illustrated in figure 6.55. For each patch, all the height constraint are identified and classified into three groups the contents of which have different roles:

- absolute and relative heights which are only found in one patch – these describe a spot height within a patch
- relative heights and offset lines with offset heights attached whose reference geometry is on a *different* patch – this describes a height at the edge of the patch with reference to another patch (see figure 5.10)
- relative heights and offset lines with offset heights attached whose reference geometry is in the *same* patch – this describes a height at the edge of the patch with reference to this patch (see figure 5.10)

Note that shared lines (where two input layers join) are treated as breaklines ('LineOffValues' with a height value of zero).

The reason for separating these heights into these three groups is because the height constraints of the first group should be prioritised for addition over those of the second group because the height will only affect that patch. The second group's height should also only affect that patch. The third group should not affect this

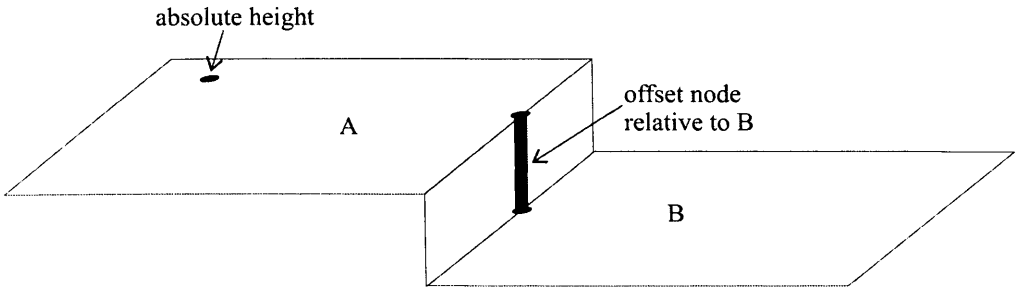


Figure 6.56: Calculating heights from surrounding patches. The relative height is with reference to polygon B, so should not affect the surface geometry of the patch containing polygon B; the surface geometry of the patch containing polygon A should be affected (have its height calculated from the patch containing polygon B). However, since the patch containing polygon B has no other height information, it instead must be used as if it is relative to polygon A.

patch because the height is described relative to this path (see section 5.3.4.3 and figure 5.10). However, *in the absence of suitable height data*, one of the geometries of the third group will be added, after which it is hopefully possible for more relative heights to be resolved. This is illustrated in figure 6.56.

Due to the inter-patch dependencies of relative heights, every addition needs to be preceded with the generation of the geometrical surface of the patch which the height is relative to, if it has not been generated already. If a reference height cannot be got, it will be skipped and the next height in the list of heights to be added will be attempted. If a height is added, the corresponding height constraint is removed from the list of heights to add and the addition of constraints is attempted from the top of the list.

6.11.3.3 Ramps and staircases

Ramps and staircases are composed of polygon(s) marked as such (a field held by 'PolygonImpl'; section 6.10.6.8). Staircase are internally stored in the same way as ramps (polygons of the areal extent) but are interpreted as being made up of evenly-spaced steps of a fixed height according to 'stepHt', a static field in 'PatchMan'.

The geometrical 3D generation of ramps and staircase shows the importance of topology. In order to render a polygon marked as a staircase, the direction of travel is needed.

The 'getMultiPatch' method which outputs a patch in 3D, automatically renders the individual steps of the staircase. For this reason, ramps and staircases are always in their own patch. In figure 6.54, the ramp is its own patch, because it is surrounded by lines which define the edges of patches (shared lines and lines bordering only one polygon – this is a requirement of all stairs and ramps). beginfigure

The topology of access across the patches composed wholly of polygons marked as ramps or stairs, is required in order to render the 3D geometry. The access centreline is generated and the 3D geometry is extrapolate on either side of this line. In order

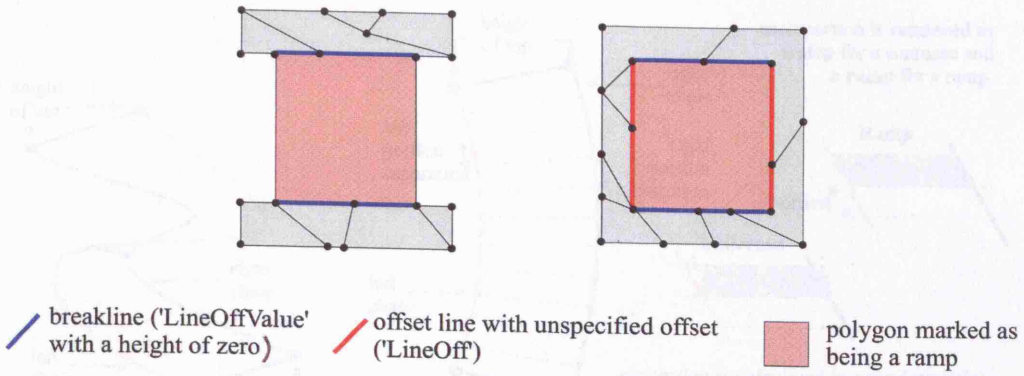


Figure 6.57: The two entry points of a polygon marked as a ramp or stair must be clear from the surrounding geometrical primitives. This figure shows that a breakline (a 'LineOffValue' with a relative height of zero) is used to indicate this. The other edges may be either not bound any other polygons of be 'LineOffs' or 'LineOffValues' with heights of greater than zero.

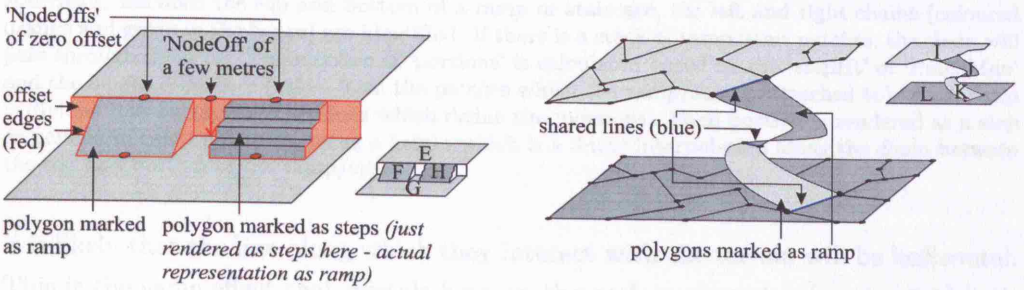


Figure 6.58: Illustration how 'inline' staircases and ramps (left) and spiral ramps and staircases (right) are split up into patches.

for the centreline to be generation, information about the two entry points is needed. Figure 6.57 shows how this is achieved and this information is used to generate a 3D geometry as shown in figure 6.59.

Figure 6.58 shows how ramps and staircases are split into patches. On the left is an inline ramp and an inline stair (within the same surface), which are surrounded by offset lines (which define the edge of patches), and offset nodes with a zero offset at the top and bottom. (Note that the offset lines at the edge of the ramps and stairs have a variable offset).

As explained earlier, a patch cannot self-intersect in two dimensions because the entire design of the model is based about the 2.5D paradigm. This may present problems for continuous spiral ramps and staircase which are very common in the built environment (e.g. fire escapes and ramps between storeys in multi-storey car parks). The solution, explained in section 5.3.3 and illustrated in figure 6.58 is to split a spiral into a stack of patches, none of which self-intersect. If this is the case, the patch along with all the other patches will be processed at the same time. Whether or not one patch is being considered or many, the procedure to rendering an output is shown in figure 6.59.

In addition, ramps and stairs affect the surface morphology of the patch by enforcing a zero gradient along the breakline. Since ramps and stairs represent access conduits,

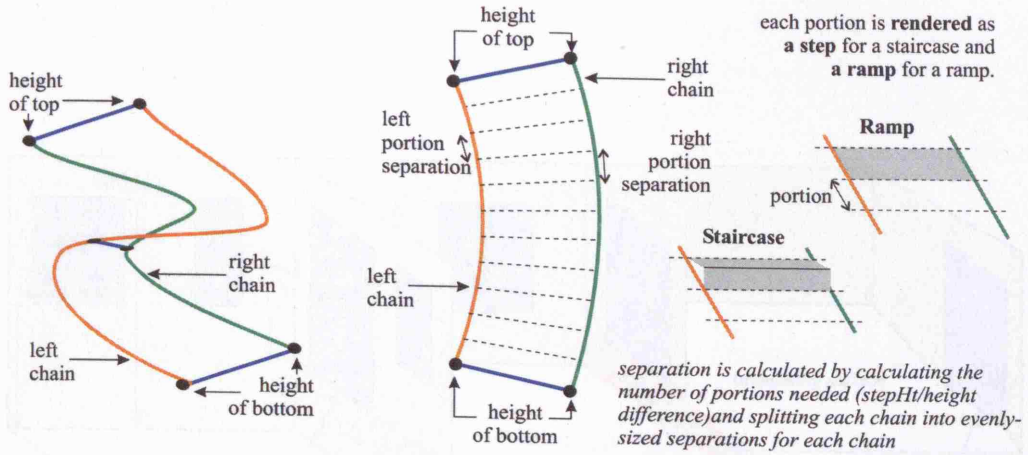


Figure 6.59: Illustration how the geometry (for Shapefile output) is calculated for ramps and staircases. Between the top and bottom of a ramp or staircase, the left and right chains (coloured orange and green in the figure) are identified. If there is a stack of ramp/stair patches, the chain will pass through them all. The number of 'portions' is calculated based on the 'stepHt' of 'PatchMan' and the height difference (taken from the patches which the ramp/stair is attached to). Each chain is divided into evenly sized sections which define the 'portions'. Each portion is rendered as a step or portion of ramp as shown, using a height which is a linear interpolation along the chain between the top and bottom of the ramp/stair.

it is likely that the line along which they interact with the terrain will be horizontal. This is the same effect that portals have on the surface geometry (section 6.11.3.4).

6.11.3.4 Patches' dependence on features

Some of the features defined in the implementation have an affect on the surface geometry of the patches; thus features can act as surface morphology information. The features which affect surface geometry are doors/windows ('Portals') and lifts ('Teleports').

'Portal' and 'Lift' features have an effect the creation of patches. 'Portals' which are flush with the ground surface effect of constraining the TIN so that it is horizontal along its length, as shown in figure 6.60. 'Lifts' have the effect of constraining the TIN so that the ground surface being horizontal in all directions.

6.11.3.5 Sensitivities to patch processing order

The order in which patches are processed to compute their surface may affect the surface geometry computed. This effect is greatest where there are few height constraints because they are likely to be more surface geometry solutions which do not violate the constraints.

This effect is illustrated in figure 6.61. Figure 6.61a shows the effect of starting from the left patch. Since it has one spot height and the 'NodeOff' cannot be resolved yet, it is assumed to be horizontal at height of 20m. The 'NodeOff' between this and the adjacent patch can now be allocated a height based on the starting patch.

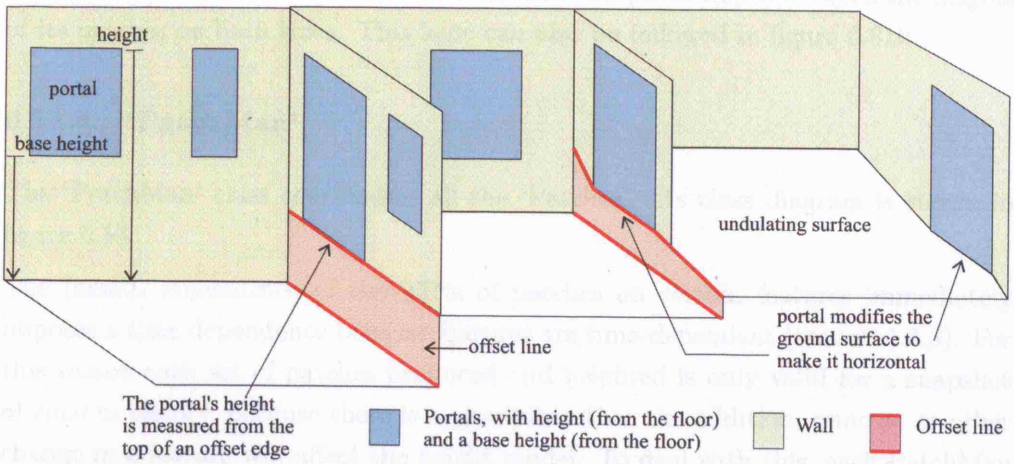


Figure 6.60: The effect that 'Doors/Windows' have on a patch's surface morphology. 'Door/Window' ('Portal') features which are flush with the ground (i.e. are likely to be doors rather than windows) have an impact on the surface geometry. Because they are likely to act as access points, it is likely that the ground is horizontal beneath. Thus some 'Door/Window' features act as pieces of surface morphology information.

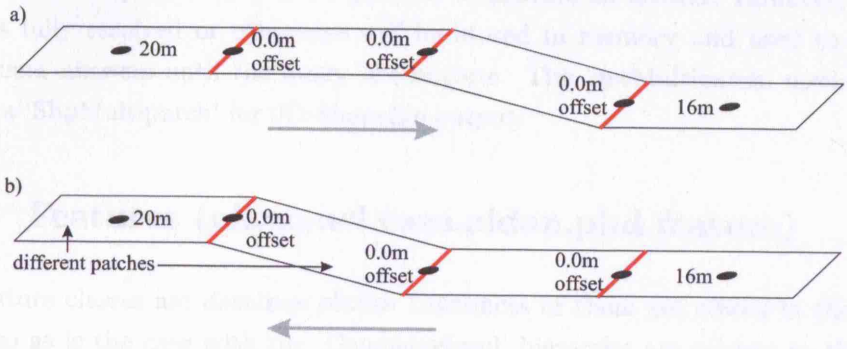


Figure 6.61: The calculation of surface morphology may be dependent on the order in which patches are visited. This figure illustrates how the patch order with little height information (the assumption that patches are horizontal has had to be used). Note that both solutions conform to the height constraints which have been imposed, but the height were delineated in different patch orders. Note that there may be many geometries in each patch.

Since the other 'NodeOff' at the edge of this patch cannot be resolved, this too is assumed to be horizontal. However, when the third patch from the left is processed, the 'NodeOff' between it and the final patch can be resolved if the final patch is assumed to be horizontal. For this reason, the third patch slopes between the heights of its patches on both sides. This logic can also be followed in figure 6.61b.

6.11.4 'PatchMan'

The 'PatchMan' class coordinates all the 'Patches'. Its class diagram is shown in figure 6.53.

The (small) dependence of the TINs of patches on certain features immediately imposes a time dependence because features are time-dependent (section 5.5.3). For this reason each set of patches produced and heighted is only valid for a snapshot of time in history, because there is a possibility that the addition, removal or other change in a feature will affect the height model. To deal with this, each PatchMan is only valid for a certain snapshot of time, along with the state of features in that snapshot.

As well as keeping the date for which its patches are valid, 'PatchMan' also keeps a list of all its patches and a reference to the instance of 'Man' (section 6.15.3) which maintains it.

Since all other components of the mapping framework do not deal with patches directly, 'PatchMan' contains methods which return heights independently of this. The public methods 'getHtHighPatch', 'getHtLowPatch' and 'getHtPatchAbove' will transparently invoke the procedure used to build and cache patches to return an answer. Thus, the first call to one of these methods from an instance of 'PatchMan' will build the required TINs of the patches to provide an answer. However, all the patches fully resolved or otherwise will be stored in memory and used to supply subsequent answers until the query is complete. The 'getMultipatch' method will return a 'ShpMultipatch' for 3D Shapefile output.

6.12 Features (uk.ac.ucl.casa.aidan.phd.feature)

The feature classes are database classes (instances of them are stored in the database), so as is the case with the 'GeometryImpl' hierarchy are subject to the conditions Ozone imposes (section 6.3). Their hierarchy is slightly different from that of the entities in the conceptual model for implementational reasons (inheritance) and in some cases they have been given different names, but as shown in table 6.4 the concepts are the same. They are not subject to the same kind of complicated arrangement as the as do the geometry classes. Instead, the simple arrangement of one set of interfaces and one set of classes are defined in the same hierarchy, in line with Ozone's requirements (section 6.3; the associated proxy and factory classes

Table 6.4: The correspondence between the 'Feature' entity hierarchy of the conceptual model and the 'FeatureImpl' class hierarchy of the logical model. The concepts are exactly the same, but in some cases are named differently

Entity	Class
Feature	FeatureImpl (abstract)
DoorKey	AccessKeyImpl
Space	SpaceImpl
Lift	TeleportImpl
BoundedSpace	BSpaceImpl
Wall	WallImpl
Door/Window	PortalImpl

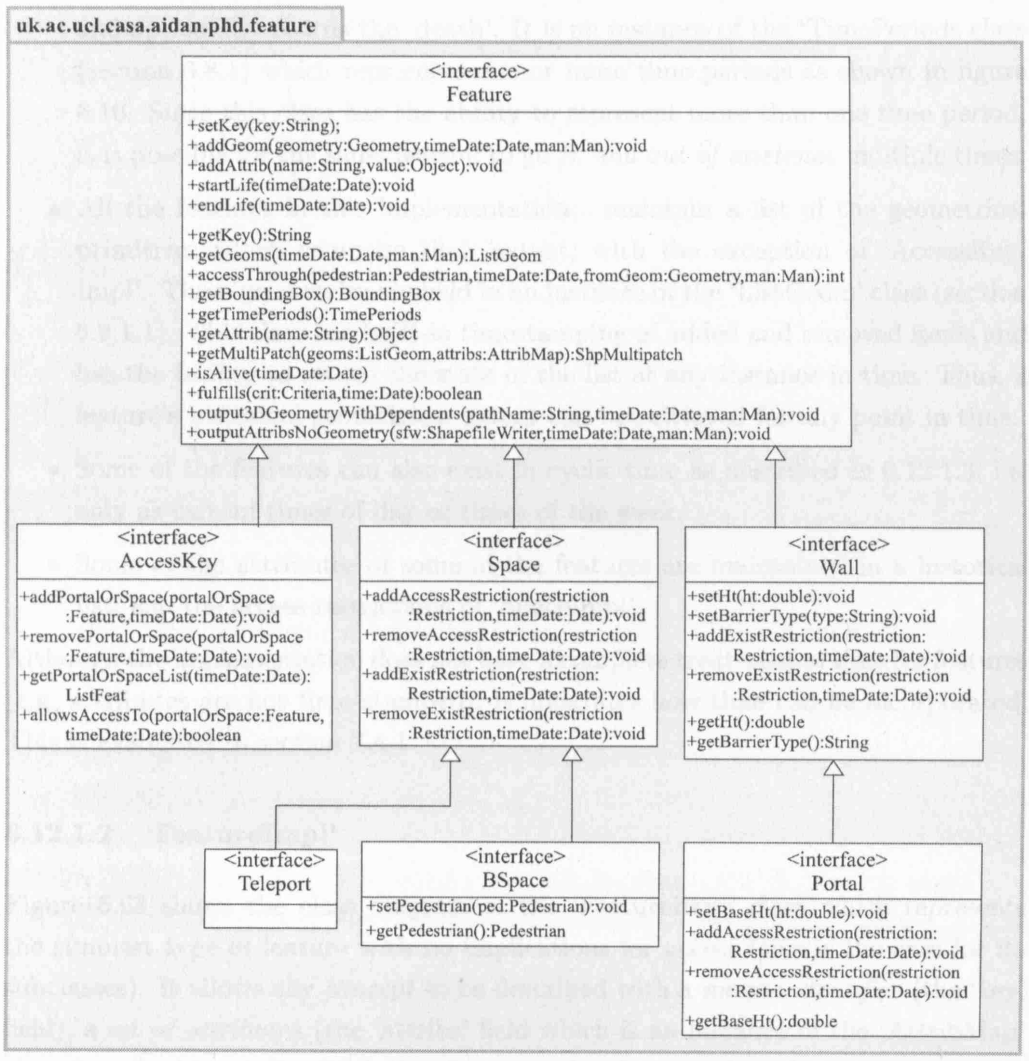


Figure 6.62: UML interface design defining the methods which should be implemented by the 'FeatureImpl' class hierarchy.

are present, but not omitted here). Figure 6.62 shows the methods defined by the 'Feature' interface hierarchy which need to be implemented by the 'FeatureImpl' class hierarchy.

6.12.1 'FeatureImpl' and its subclasses

'FeatureImpl' and its subclasses are arranged in the same hierarchy as their interfaces (this is also the case for 'GeometryImpl'; a requirement of database class in Ozone).

6.12.1.1 Time

Time is incorporated in the feature description in various ways:

- The 'timePeriods' field (see figure 6.63) represents the time period over which the feature is in existence. It will always records the 'birth' of the feature and optionally records the 'death'. It is an instance of the 'TimePeriods class (section 6.8.1) which represents one or more time periods as shown in figure 6.10. Since this class has the ability to represent more than one time period, it is possible for the same feature to go *in and out of existence* multiple times.
- All the features in this implementation maintain a list of the geometrical primitives which comprise their extent, with the exception of 'AccessKeyImpl'. These geometries are held in an instance of the 'ListGeom' class (section 6.9.1.1). This class has built-in timestamping of added and removed items and has the facility to return the state of the list at any instance in time. Thus, a feature's historical geometrical extent can be retrieved for any point in time.
- Some of the features can also exist in cyclic time as described in 6.12.1.3; i.e. only as certain times of day or times of the week.
- Some of the attributes of some of the features are maintained in a historical list; e.g. the access restrictions of 'SpaceImpl'.

Although the implementation does not have a complete treatment of time for features (e.g. attributes are not time-stamped), it illustrates how time can be incorporated. This is evaluated in section 7.4.1.

6.12.1.2 'FeatureImpl'

Figure 6.63 shows the class diagram of the 'FeatureImpl' class which represents the simplest type of feature with no implications for access (this is the case for its subclasses). It allows any *concept* to be described with a *unique identifier* (the 'key' field), a set of *attributes* (the 'attribs' field which is an instance of the 'AttribMap' class described in section 6.9.1.4) and a *geometrical extent* described by any number of 'Geometries' (stored in the 'list' field).

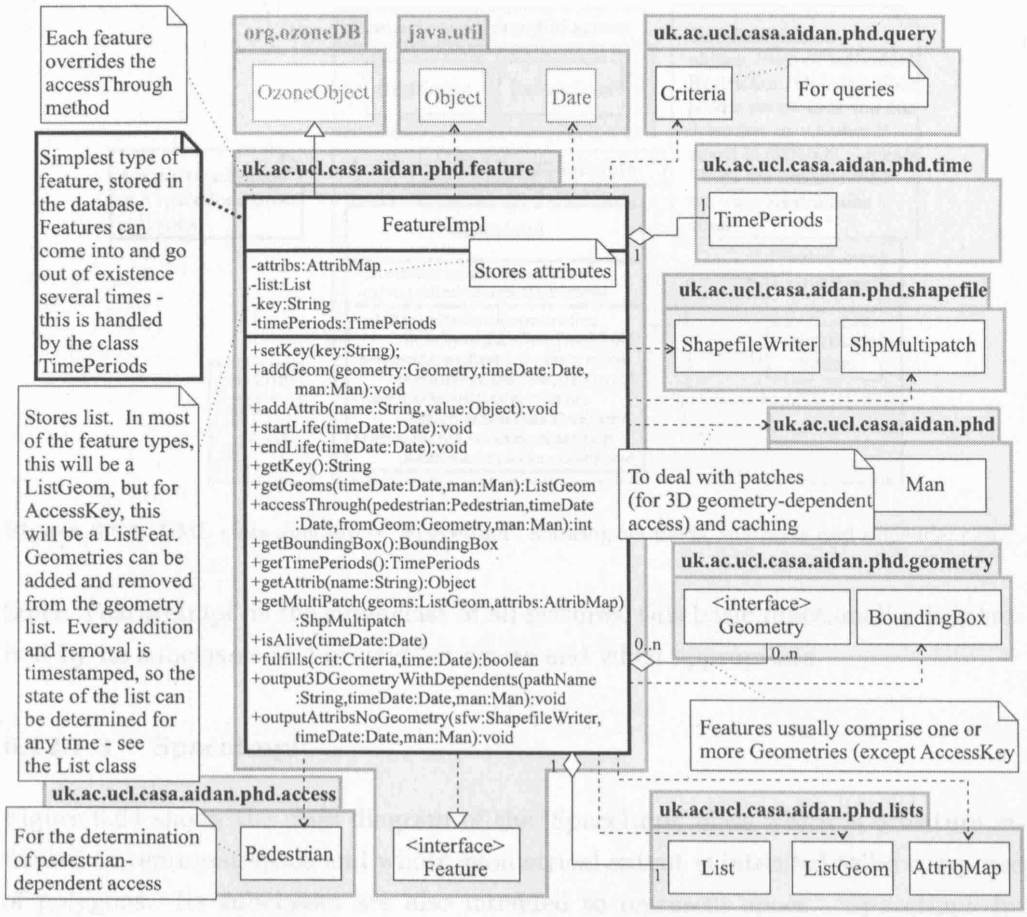


Figure 6.63: UML class diagram of 'FeatureImpl', showing its fields, methods and dependencies.

The 'accessThrough' method for 'FeatureImpl' which assesses whether access is allowed (the method may be called by the method of the same name of one of its geometries) always allows access (because this class does not affect access). This method is overridden by its subclasses (many of which *do* affect access).

This class and its subclasses also have the 'output3DGeometryWithDependents' method which writes themselves to 3D Shapefiles (section 6.7) along with the other features which share parts of the geometry. It uses the classes in the 'uk.ac.ucl.casa.aidan.phd.geometry.threedimensional' (section 6.11) to 'reason' (interpolate/extrapolate) the 3D geometry, before outputting. Since the 'uk.ac.ucl.casa.aidan.phd.geometry' package only deals with connected surfaces in 3D including staircases and ramps, any 3D geometry of structures above the ground is parameterised and held by the feature. Since 'FeatureImpl' is the most basic type of feature and has none of its geometry parameterised, the 'output3DGeometryWithDependents' simply output the nodes, lines and polygons which comprise its geometrical extent, with the height taken from the patches generated by 'PatchMan', along with all the other features which shared some or all of the geometries.

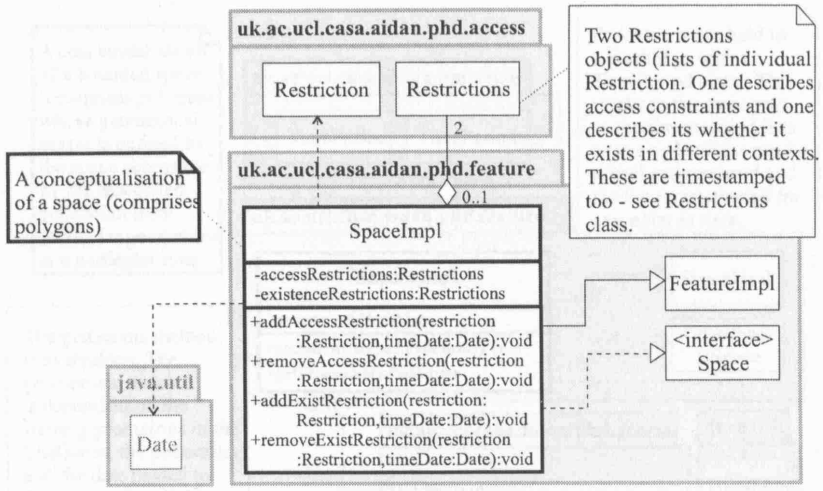


Figure 6.64: UML class diagram of 'SpaceImpl', showing its fields, methods and dependencies.

Since 'FeatureImpl' is the superclass of all features, this basic functionality is inherited by its subclasses and overridden where and when appropriate.

6.12.1.3 'SpaceImpl'

Figure 6.64 shows the class diagram of the 'SpaceImpl' class which is a feature intended to represent space and whose geometrical extent is intended to be composed of polygons. Its subclasses are also intended to represent space. 'SpaceImpl' inherits all the fields and methods of 'FeatureImpl', but adds 'accessRestrictions' and 'existenceRestrictions' fields which are instances of the 'Restrictions' class (section 6.14.2.2). The 'accessRestrictions' attribute uses the 'Restrictions' class to describe the time- and pedestrian-dependent conditions for which a 'Pedestrian' can gain access to the space. This has direct implications for the 'accessThrough method'. The 'existenceRestrictions' attribute uses the same method to describe whether the feature is in existence in cyclic time, as described in section 5.5.3.

The 'AccessResolver' entity (section 5.7) does not match to a class. Instead, its functionality is implemented in the 'accessThrough' method. The version in 'FeatureImpl' is overridden to take into account of whether the feature is physically present at the instant in time (if not, access is not hindered) and if it is, whether the 'accessRestrictions' are fulfilled. In summary, the 'accessThrough' method for 'SpaceImpl' takes into account:

- the lifetime of the feature (birth and death)
- its 2D geometrical extent at the time
- whether it is in existence in cyclic time (section 5.5.3)
- how the characteristics of the pedestrian and the time fulfil the access restrictions – this is the core of the access model as described in section 5.7

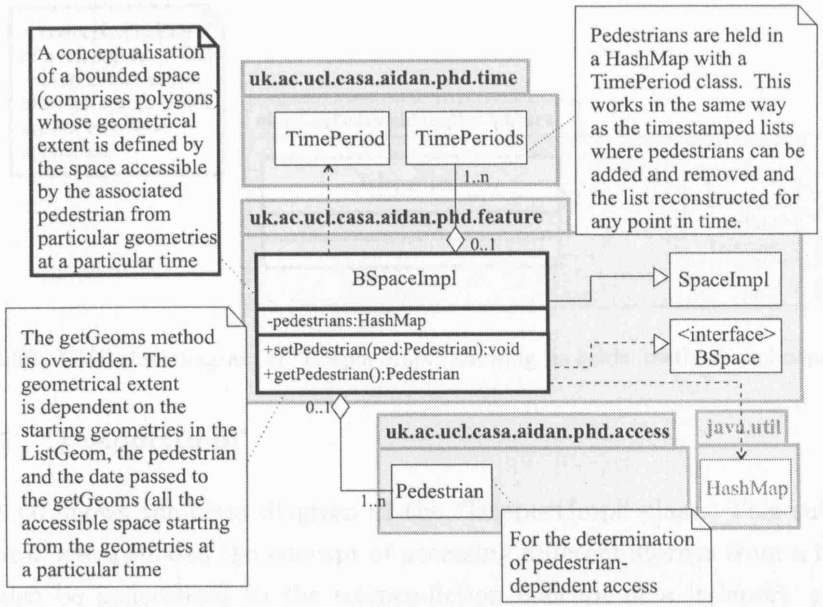


Figure 6.65: UML class diagram of 'BSpaceImpl', showing its fields, methods and dependencies.

The 'output3DGeometryWithDependents' is inherited from 'FeatureImpl', unchanged.

6.12.1.4 'BSpaceImpl'

Figure 6.65 shows the class diagram of the 'BSpaceImpl' class (which corresponds to the 'BoundedSpace' entity). This subclass of 'SpaceImpl' differs in one respect: its geometrical extent is pedestrian-dependent (section 5.5.1.2). Using the instance of its 'Pedestrian' attribute and starting from the geometry or geometries in the geometry list, the 'accessThrough' method recursively collects all the spaces to which the pedestrian has access using one of the 'getAdjRecurs' methods.

The 'pedestrians' field is a 'HashMap' with pedestrians (section 6.14.1) keyed to 'TimePeriods'. In exactly the same way as how 'List' manages the times at which items are added and removed, 'BSpaceImpl' manages changes in the 'Pedestrian' which defines its geometrical extent. Thus, when the geometrical extent is queried, the extent at different time snapshots can be delineated.

Since the extent is dependent on accessibility and accessibility constraints may change through time (access restrictions and geometrical extents of spaces and barriers), the extent is dependent on the interaction of the surrounding features in their state as a particular snapshot.

The 'output3DGeometryWithDependents' is inherited from 'SpaceImpl', unchanged. Due to the access-based geometrical extent, this is likely to be surrounded by 'Wall' features (which are added by virtue of them sharing some of the geometries of the 'BSpace' feature).

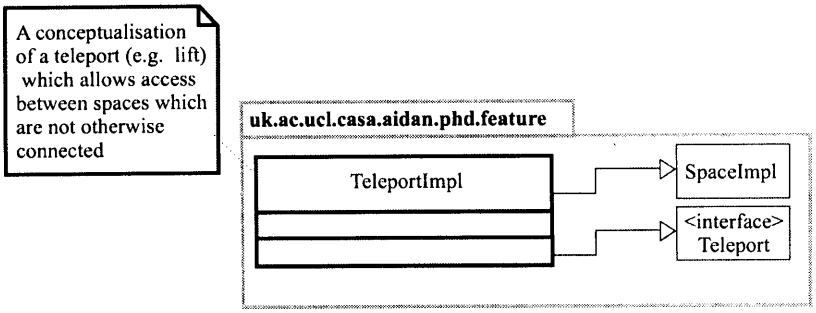


Figure 6.66: UML class diagram of 'TeleportImpl', showing its fields, methods and dependencies.

6.12.1.5 'TeleportImpl'

Figure 6.66 shows the class diagram of the 'TeleportImpl' class. This subclass of 'SpaceImpl' encapsulates the concept of accessing different storeys from a lift shaft. It can also be generalised to the science-fiction concept of a 'teleport' providing access between different polygons (because polygons need not necessarily be in a vertical stack).

A 'TeleportImpl' is similar to a 'SpaceImpl'. It comprises the footprints of the lift shaft on each storey. The only difference in operation from its subclass is that its 'getAdjRecurs' method, when applied to access, will return all the polygons of the teleport. Thus, if access is gained to any footprint of the lift, all the other footprints of the lift shaft are accessible, each one being topologically connected to adjacent polygons on their respective storey. The lift shaft is usually surrounded by walls and at least one door on each storey.

The 'Teleport' feature is illustrated in figure 6.67. Access to the footprint of a lift shaft (coloured dark red) is dependent on the access restrictions inherited from 'SpaceImpl' and provided by the 'Portal' feature adjacent to the 'Teleport' on each storey.

The 'Teleport' feature may also affect the surface geometry of the patch it is in, such that its footprint on each storey is horizontal (section 6.11.3.4).

6.12.1.6 'Wall'

This class and its subclass represent barriers to pedestrian movement in the built environment. Figure 6.68 shows the class diagram of the 'WallImpl' class.

The 'WallImpl' class maintains a static list of 'barrierTypes', listed in table 5.2 which indicates the ease with which it can be breached for access. Each instance holds a value for itself in the 'barrierTypeID' field (which also inherited by 'Portal'). A 'Pedestrian' can breach the barrier if its 'maxBreachLevel' (the security level the pedestrian is willing or able to breach) is greater than that of the 'Wall' or 'Portal', as described in section 5.7.1.4. 'Walls' also have a height attribute and

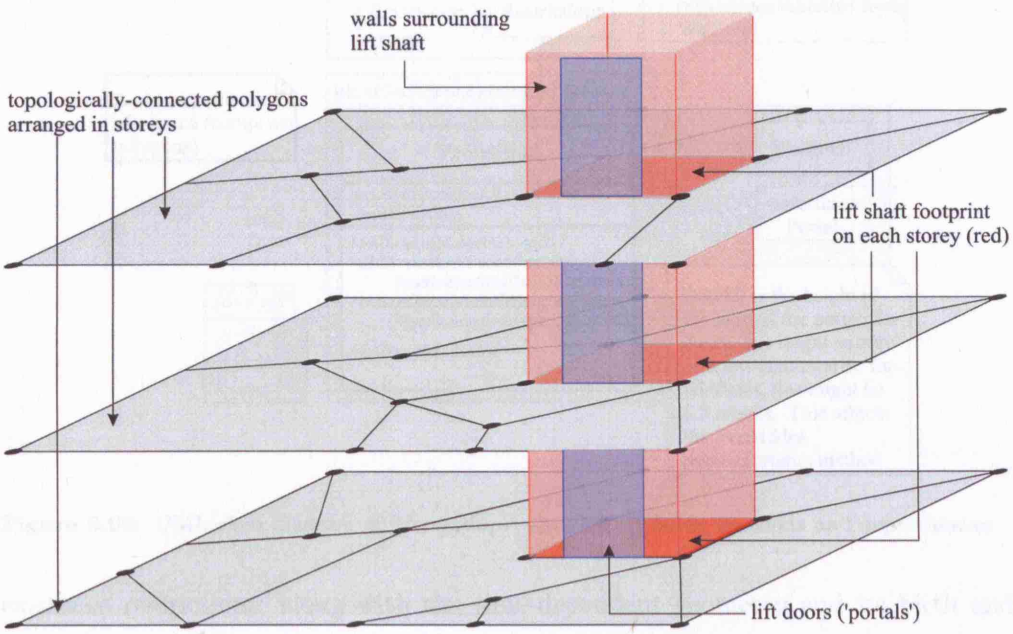


Figure 6.67: Illustration of a 'Teleport'. The 'TeleportImpl' feature is a collection of polygons (shown in dark red) representing the footprint of the teleport (lift in this case) on the storey. These footprints are connected to the other polygons on the same storey in the usual way. The lift is surrounded by 'Wall' features and there is at least one 'Portal' feature on each storey.

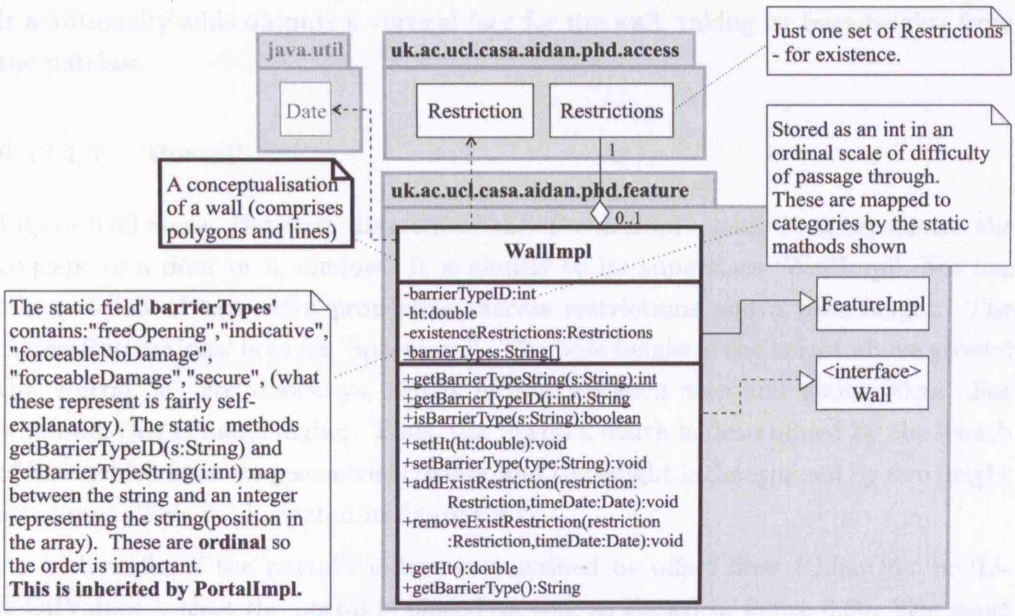


Figure 6.68: UML class diagram of 'WallImpl', showing its fields, methods and dependencies.

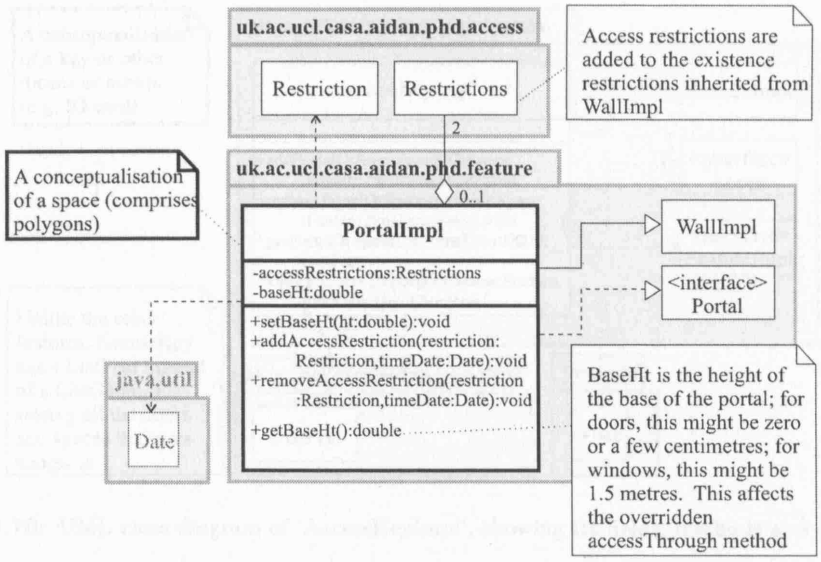


Figure 6.69: UML class diagram of 'PortalImpl', showing its fields, methods and dependencies.

existence restrictions; along with the time-dependent geometry and its birth and death (inherited from 'FeatureImpl') affect access.

If the wall's extent is described by offset lines ('LineOffs' or 'LineOffValues'), then the wall is placed on upper surface, as shown in figure 6.60.

Since the geometry of 'WallImpl' extends above the ground surface, its vertical extent is parameterised by its 'ht' attribute which describes the relative height from the ground level. If there is a patch above, the height will instead be taken from this. The 'output3DGeometryWithDependents' is inherited from 'FeatureImpl', but it additionally adds outputs a vertical face for the wall, taking its base heights from the patches.

6.12.1.7 'Portal'

Figure 6.69 shows the class diagram of the 'PortalImpl' class. It encapsulates the concept of a door or a window. It is similar to its superclass 'WallImpl', but has the addition of two extra properties: access restrictions and a base height. The 'accessRestrictions' is as for 'SpaceImpl'. The base height is the height above ground the 'Portal' is. For doorways, this is usually between zero and about 30cm. For windows, this is much higher. Thus, the Portal's width is determined by the length of lines which form its geometrical extent and the height is determined by two height attributes. This is illustrated in figure 6.60.

As with walls, if the portal's extent is described by offset lines ('LineOffs' or 'LineOffValues'), then the portal is placed on top, as shown in figure 6.60. The same figure also shows that the portal may affect the surface geometry of the terrain, as explained in section 6.11.3.4

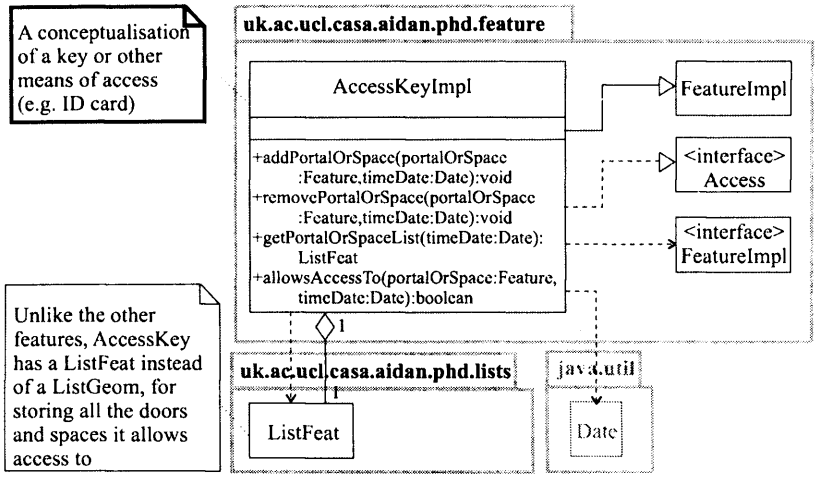


Figure 6.70: UML class diagram of 'AccessKeyImpl', showing its fields, methods and dependencies.

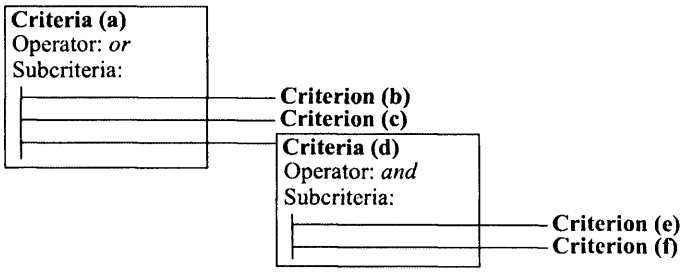


Figure 6.71: The 'Criteria' class encapsulates a nesting in a structured nested query. Each 'Criteria' has an operator ('and' or 'or' and a list of either other instances of 'Criteria' or 'Criterion'. The example above equates to: *b or c or (e and f)*.

The 'output3DGeometryWithDependents' is inherited from 'PortalImpl', but it also deals with the additional 'baseHt' attribute described above.

6.12.1.8 'AccessKey'

Figure 6.70 shows the class diagram of the 'AccessKeyImpl' class, which encapsulates the concept of an access key (corresponding to the 'DoorKey' entity).

6.13 Queries (uk.ac.ucl.casa.aidan.phd.query)

A query language has been designed and implemented. There are two main uses:

- by subclasses of List (section 6.9.1.1) for retrieving data
- by the 'Restriction' class (section 6.14.2.1)

The class 'Criteria' encapsulates the concept of a query composed of a nested set of expressions containing either 'Criteria' or 'Criterion' classes, combined with the 'and' and 'or' operators, as illustrated by figure 6.71. The abstract class 'Criterion' represents specific conditions (e.g. 'age > 23'). It has three concrete subclasses

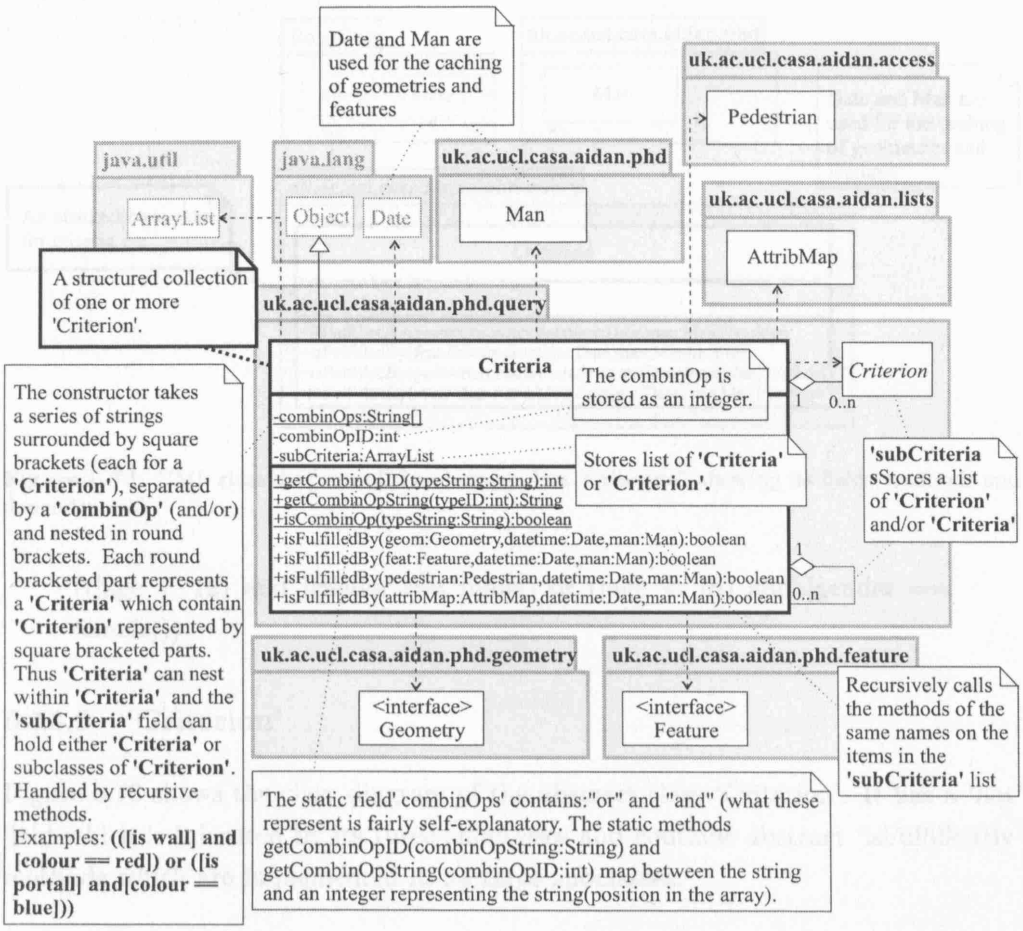


Figure 6.72: UML class diagram of 'Criteria', showing its fields, methods and dependencies.

('CritType', 'CritAttrib' and 'CritAccessKey' which encapsulate different types of expressions.

6.13.1 'Criteria'

Figure 6.72 shows the class diagram of the class 'Criteria'. 'Criteria' has a number of static methods which define the operators (currently 'and' and 'or'), one of which values is held in the private 'combinOpID' field. This operator applies to all the 'Criteria' or 'Criterion' listed in the 'subCriteria' field; thus one 'Criteria' may contain nested 'Criteria' (methods of which are called recursively) and various types of 'Criterion'. The methods 'isFulfilledBy' take various parameters and return true or false, depending on whether the 'Geometry', 'Feature', 'Pedestrian' or 'AttribMap' is fulfilled by the 'Criteria'

Although not shown, the 'Criteria' class' constructor takes a 'String' and builds a representation of the meaning of the 'String'. Individual 'Criteria' are contained in round brackets, and individual 'Criterion' are contained in square brackets; for example:

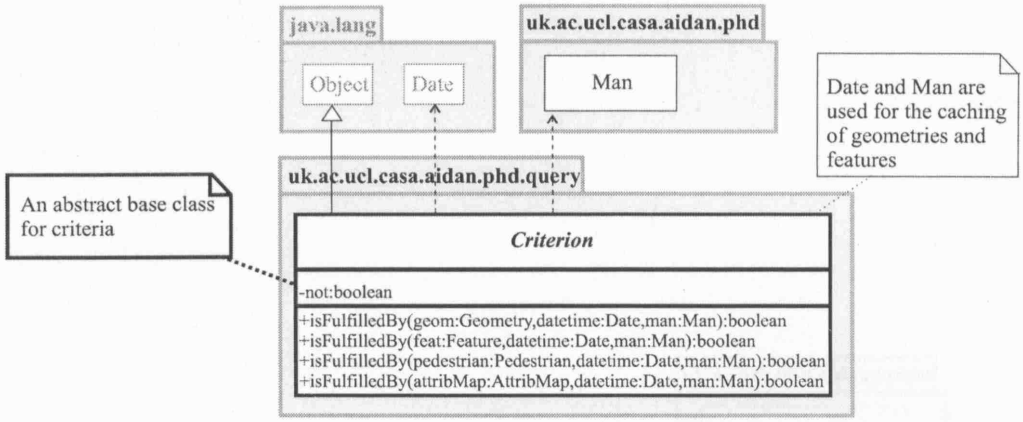


Figure 6.73: UML class diagram of the abstract class 'Criterion', showing its fields, methods and dependencies.

(([age < 12) and [gender == male]) or ([age < 16) and [gender == female]))

6.13.2 'Criterion'

Figure 6.73 shows the class diagram of the abstract class 'Criterion'. It has a 'not' field which is inherited by its three subclasses and contains abstract 'isFulfilledBy' methods which are implemented in its three subclasses.

6.13.2.1 'CritType'

Figure 6.74 shows the class diagram of the class 'CritType', which represents a type of Geometry or Feature. The static field 'types' holds all the possible values which are shown at the bottom of figure 6.74. The static methods can turn these into a numerical value, and this is what is stored in the 'type' field. The 'isFulfilledBy' methods compare the object passed to it with the 'Criterion' is encodes (not all objects are relevant; e.g. 'Pedestrian').

'CritType' is often part of the 'Criteria' for the 'ListGeomCrit' and 'ListFeatCrit' classes (sections 6.9.1.2 and 6.9.1.3). For example, if a 'ListGeomCrit' has 'Criteria' set as '([is line])', then it will only accept lines to be added in the list.

6.13.2.2 'CritAttrib'

Figure 6.75 shows the class diagram of the class 'CritAttrib', which represents an attribute value which is relevant for a 'Feature' an 'AttribMap' or a 'Pedestrian', all of which have (or are) 'AttribMaps'. It allows the value of an attribute to be encoded, using the operators shown at the bottom of figure 6.75.

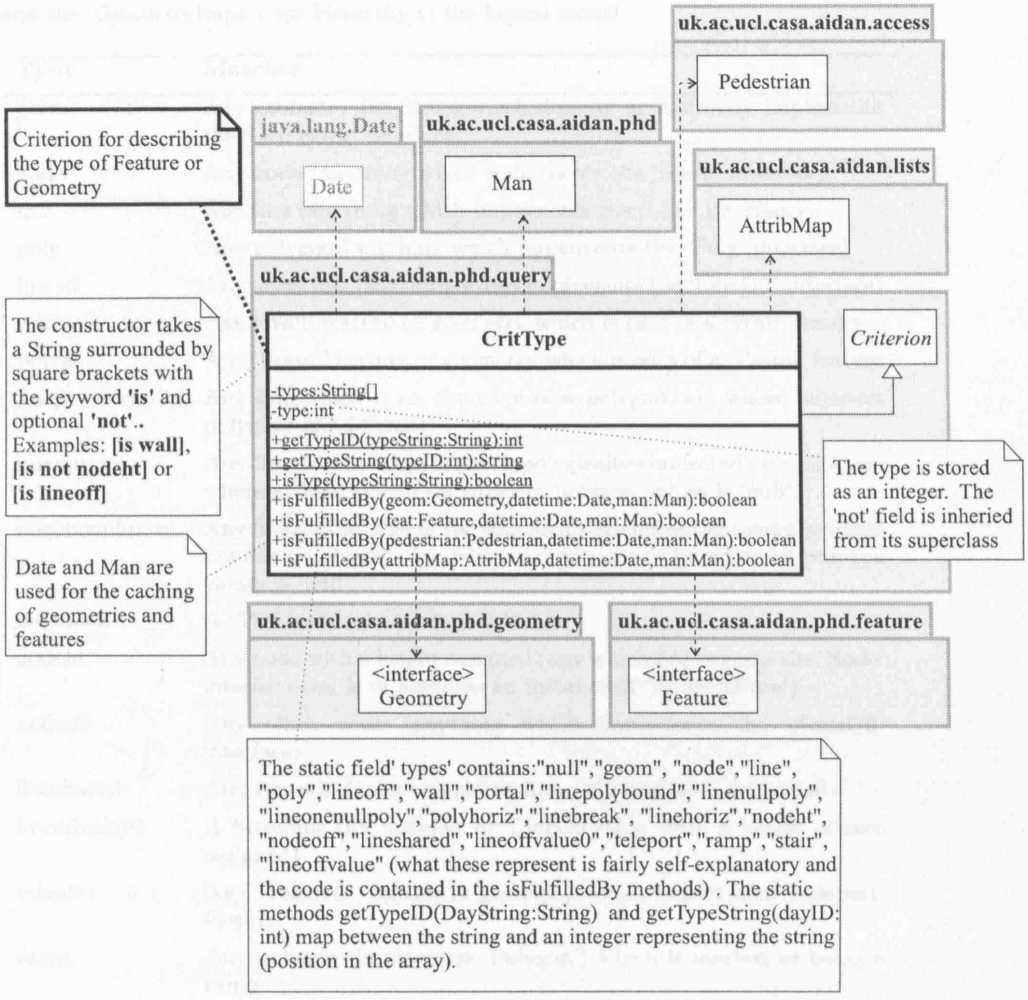


Figure 6.74: UML class diagram of the class 'CritType', showing its fields, methods and dependencies.

Table 6.5: The correspondence between the 'Geometry' entity hierarchy of the conceptual model and the 'GeometryImpl' class hierarchy of the logical model.

Type	Matches
geom	Any geometry (anything which directly or indirectly implements the 'Geometry' interface)
node	Any node (anything which implements the 'Node' interface)
line	Any line (anything which implements the 'Line' interface)
poly	Any polygon (anything which implements the 'Poly' interface)
lineoff	Any offset line (anything which implements the 'LineOff' interface)
wall	Any 'Wall' feature or geometry which is part of a 'Wall' feature
portal	Any 'Portal' feature or geometry which is part of a 'Portal' feature
linepolybound	Any line which is on the edge of a polygon, i.e. whose adjacent polygons are different)
linenullpoly	Any line on the edge of the topologically-connected coverage, i.e. where at least one of its adjacent polygon values is 'null'.
lineonenullpoly	Any line on the edge of the topologically-connected coverage which bounds a polygon , i.e. where exactly one of its adjacent polygon values is 'null'.
linebreak	A 'TinLineBreak' (section 6.11.1).
nodeht	Any node with a height assigned (any which implements the 'Node' interface and is or contains an instance of 'NodeHtPure').
nodeoff	Any offset node (anything which implements the 'NodeOff' interface)
lineshared	Any line which can coincident lines (sharing both its nodes)
lincoffvalue0	A breakline (an instance of 'LineOffValue' with a height of zero assigned)
teleport	Any 'Teleport' feature or geometry which is part of a 'Teleport' feature
ramp	Any polygon (implements 'Polygon') which is marked as being a ramp
stair	Any polygon (implements 'Polygon') which is marked as being stairs
lineoffvalue	An offset line with a height assigned (an instance of 'LineOffValue')

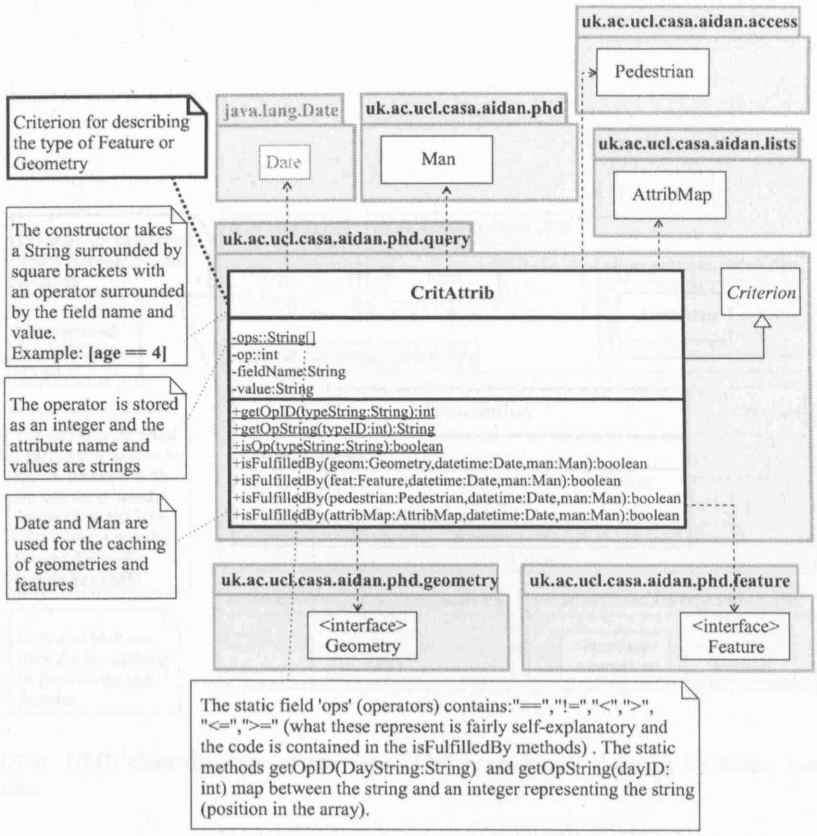


Figure 6.75: UML class diagram of the class 'CritAttrib', showing its fields, methods and dependencies.

6.13.2.3 'CritAccessKey'

Figure 6.76 shows the class diagram of the class 'CritAccessKey'. It is only relevant for a 'Pedestrian' and is used to describe the 'AccessKeys' (section 6.12.1.8) a pedestrian is in possession of. It is used by the 'Restriction' class (section 6.14.2.1).

6.14 Access (uk.ac.ucl.casa.aidan.phd.access)

This package contains three classes modelling data structures for the use of the access delineation routines:

- 'Pedestrian' (section 6.14.1) – a data structure representing a pedestrian.
- 'Restriction' (section 6.14.2.1) – a data structure representing a time- and pedestrian-dependent access restriction.
- 'Restrictions' (section 6.14.2.2) – a data structure representing a timestamped list of instances of 'Restriction'.

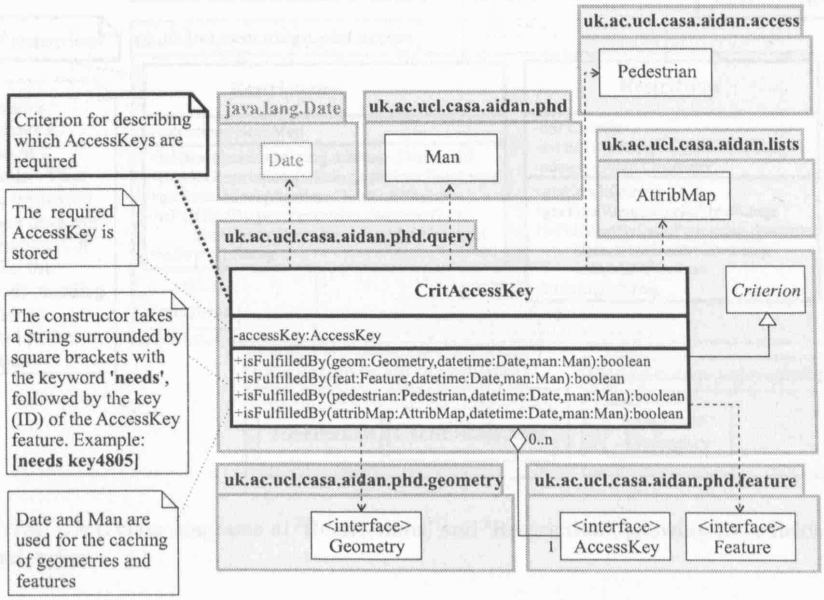


Figure 6.76: UML class diagram of the class 'CritAccessKey', showing its fields, methods and dependencies.

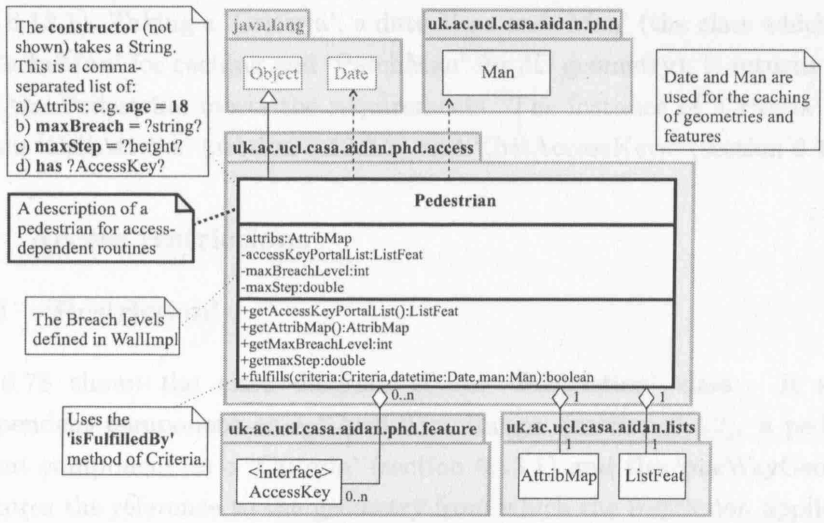


Figure 6.77: UML class diagram of 'Pedestrian', showing its fields, methods and dependencies.

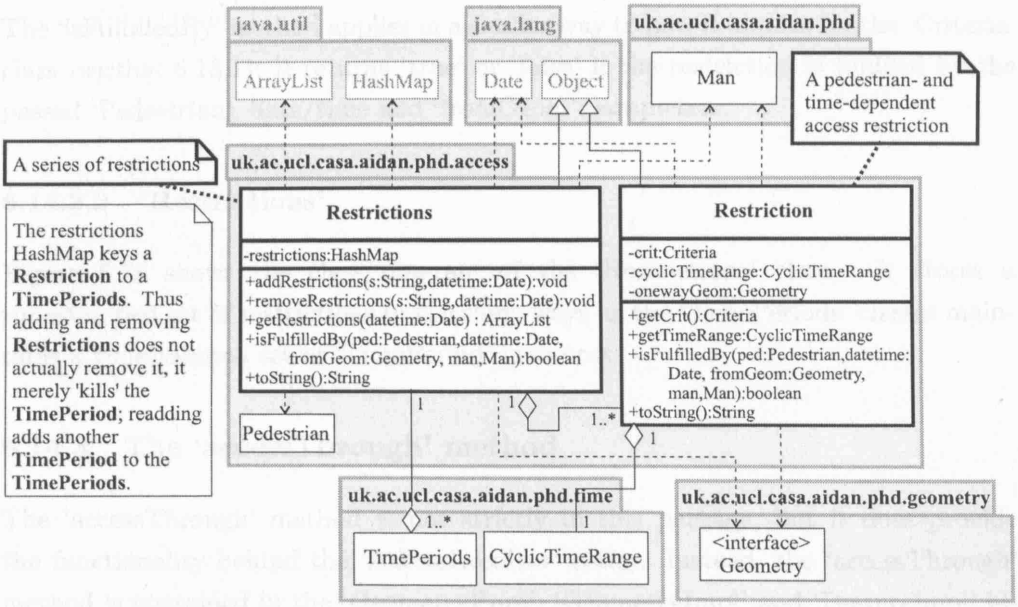


Figure 6.78: UML class diagrams of 'Restrictions' and 'Restriction', showing their fields, methods and dependencies.

6.14.1 'Pedestrian'

Figure 6.77 shows the class diagram of the 'Pedestrian' class. Its fields store a set of attributes, a list of door keys in possession and individual doors through which a pedestrian has access, the maximum security level which the pedestrian is willing or able to breach and the maximum height of step which the pedestrian is able to negotiate. The significance of these are explained in section 5.7.1.1.

The 'fulfills' method works in a similar way to the 'isFulfilledBy' method of 'Criteria' (section 6.13.1). Taking a 'Criteria', a date/time and 'Man' (the class which coordinates 'CacheMan' for caching and 'PatchMan' for 3D geometry), it returns 'true' or 'false' if the Pedestrian meets the requirements. The instance of 'Criteria' is likely to contain 'CritAttribs' (section 6.13.2.2) and 'CritAccessKeys' (section 6.13.2.3).

6.14.2 Access restrictions

6.14.2.1 'Restriction'

Figure 6.78 shows the class diagram of the 'Restriction' class. It stores a time-dependent component as a 'CyclicTimeRange' (section 6.8.2), a pedestrian-dependent component as a 'Criteria' (section 6.13.1) and the 'oneWayGeom' field which stores the reference to the geometry from which the restriction applies. This latter field is only used if the restriction applies one way and applies to a specific feature which is adjacent to the referenced geometry. These fields are according to section 5.7.1.4.

The 'isFulfilledBy' method applies in a similar way to how it applies for the 'Criteria' class (section 6.13.1); it returns 'true' or 'false' if the restriction is fulfilled by the passed 'Pedestrian', date/time and 'fromGeom' parameters.

6.14.2.2 'Restrictions'

Figure 6.78 shows the class diagram of the 'Restrictions' class. It stores a timestamped set of restrictions in the same ways as the 'TimePeriods' classes maintains a timestamped set of 'TimePeriod' instances.

6.14.3 The 'accessThrough' method

The 'accessThrough' method is not strictly in this package, but it does provide the functionality behind the 'AccessResolver' entity. Instead, the 'accessThrough' method is contained in the 'GeometryPure', 'GeometryImpl' and 'FeatureImpl' hierarchies. In the 'GeometryPure' hierarchy, it is passed straight through to the 'GeometryImpl' hierarchy. In the 'FeatureImpl' is assesses whether access is allowed through that feature.

The main use is through that of the 'GeometryImpl' hierarchy. As seen in figure 6.79, this first accesses whether access is geometrically allowed and then tests every feature whose extent incorporates the geometry. Access is geometrically allowed if the step height and gradient are below the pedestrian's threshold (for this reason, the '3DReasoner' must be called). Access through a feature is allowed according to the specific feature class' 'accessThrough' method. This returns an integer indicating the ease of access, which is passed back to the geometry. The geometry eventually returns the value of the easiest access.

Ease of access is only an issue for the features which act as barriers which are 'Walls' and 'Portals'. Each of these has a 'barrierTypeID' field which indicates the physical strength of the barrier for impeding access according to the ordinal scale shown in table 5.2. If access is not allowed, the feature will be assessed as to whether it is weaker than the strength of barrier which the pedestrian is able or willing to breach barrier, through its 'maxBreach' attribute. The scale of ease is the same scale of the barrier strength scale; it is equal to the strength of the barrier that was breached. For example, if a pedestrian can breach 'secure' and a wall is 'secure', then the ease of access (actually, the difficulty of access) is 4 (table 5.2; whereas if the wall was 'forceableNoDamage' then the ease (difficulty) of access would be 2. The resulting value of this from the 'accessThrough' method of 'GeometryImpl' will indicate whether a barrier had to be breached and how difficult this was. Also most pedestrian would not be able to breach a 'secure' barrier, breaching 'indicative' barriers can be considered to be common practice.

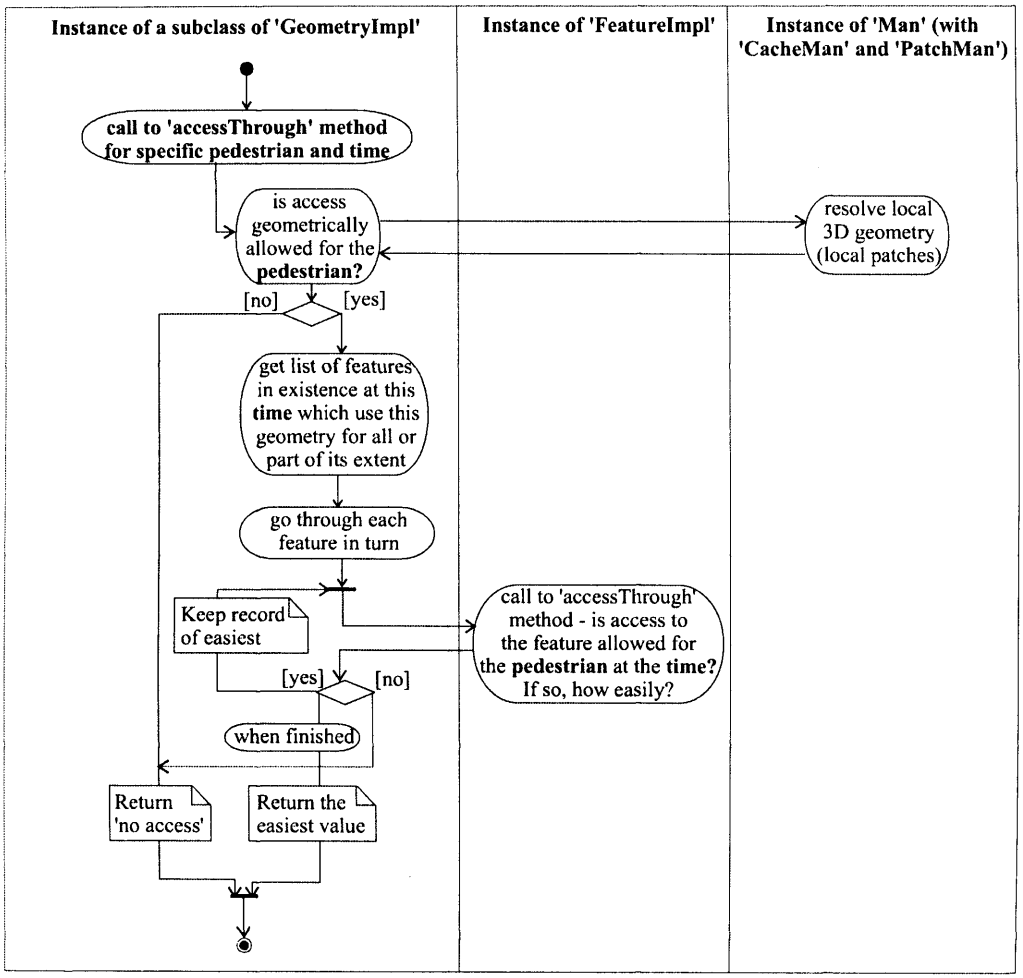


Figure 6.79: UML sequence diagram showing the use of the 'accessThrough' method on a subclass of 'GeometryImpl'. The returned value is an integer which indicates whether access is allowed and if it is allowed, indicates the ease of access.

6.15 Caching (uk.ac.ucl.casa.aidan.phd.cache)

This package contains three classes for caching specific instances of classes ('GeometryPure', 'PolygonPure' and 'BSpaceImpl'), 'CacheMan' which coordinates the caching for a particular date/time and 'Man' which coordinates 'CacheMans' and 'PatchMans' (section 6.11) for various time/dates.

'GeometryPure', 'PolygonPure' and 'BSpaceImpl' are the only classes which are cacheable (it is considered that the other classes do not need to be).

It is important to note that *all* the caching is done in memory only and none of this is stored in the database. Thus, the caching persists only as long as the software application is running.

- 'Man' (section 6.15.1) – manages and coordinates 'CacheMans' and 'PatchMans' (section 6.11) for different date/times.
- 'CacheMan' (section 6.15.1) – manages and coordinates the caching for a particular date/time.
- 'GeometryCached' (section 6.15.2.1) – deals with caching 'Geometries'.
- 'PolygonCached' (section 6.15.2.2) – deals with caching 'Polygons'/
- 'BSpaceCached' (section 6.15.2.3) – deals with caching 'BSpaces' (other features do not need to be cached.

Figure 6.80 shows how 'CacheMan' interacts with the cached objects and their counterparts. 'CacheMan' is responsible for maintaining a list of the existing cached objects, which are valid for a particular date/time; results may be time-dependent. The corresponding class ('GeometryPure' in the case of 'GeometryCached') requests a cached version of the object (for a particular date/time) from 'CacheMan'. 'CacheMan' will then return the cached version if it exists, otherwise it will create a cached version with no stored information. If the cached version has a stored answer, this will be used; if not then an answer will be calculated, stored in the cached version and returned.

6.15.1 'CacheMan'

Figure 6.81 shows the class diagram of the 'CacheMan' class. Each instance of 'CacheMan' is linked to a specific date/time (the different 'CacheMans') are coordinated by the 'Man' class (section 6.15.3). An instance of 'CacheMan' can return a cached version of any cacheable object ('GeometryPure', 'PolygonPure' or 'BSpaceImpl'), returning the stored one (held in the 'HashMap' field) or creating a new one, storing and returning it. Cached versions of cacheable objects can also be removed.

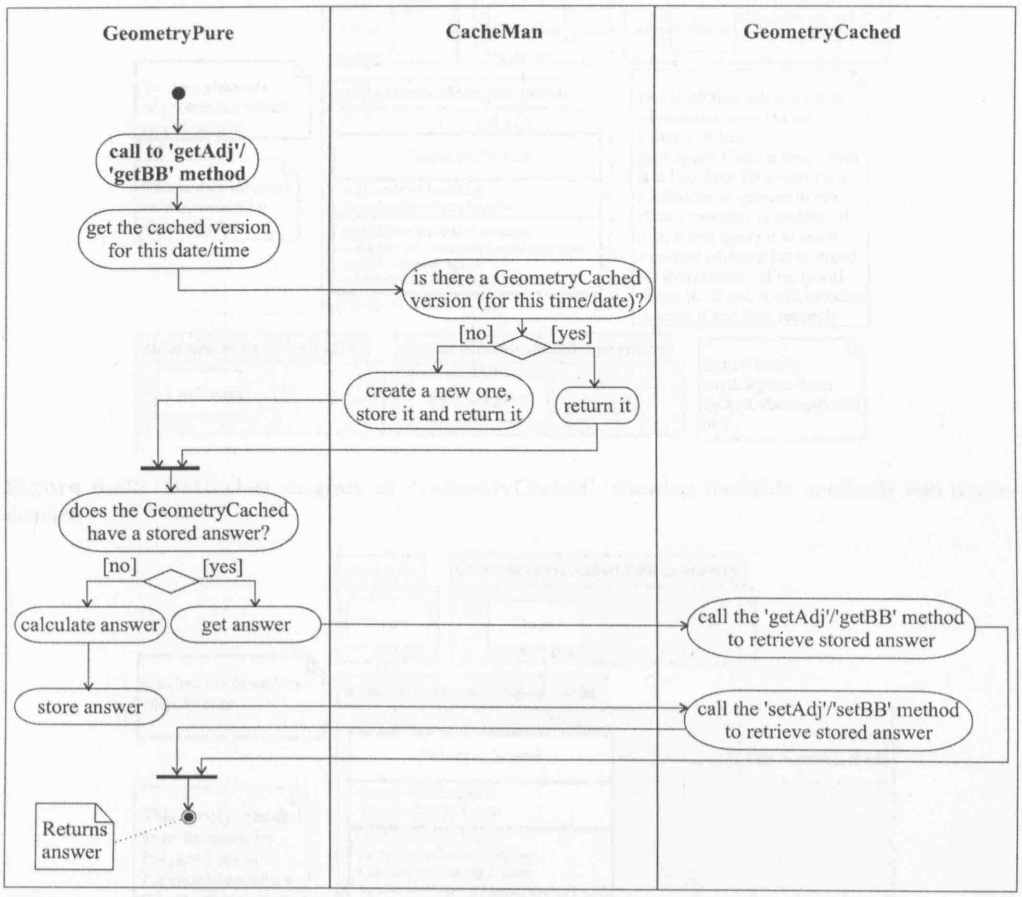


Figure 6.80: UML sequence diagram showing an object using its cached object counterpart (GeometryPure–GeometryCached, PolygonPure–PolygonCached, BSpaceImpl–BSpaceCached). This procedure is the same for all the pairs of cacheable objects and their cached counterparts.

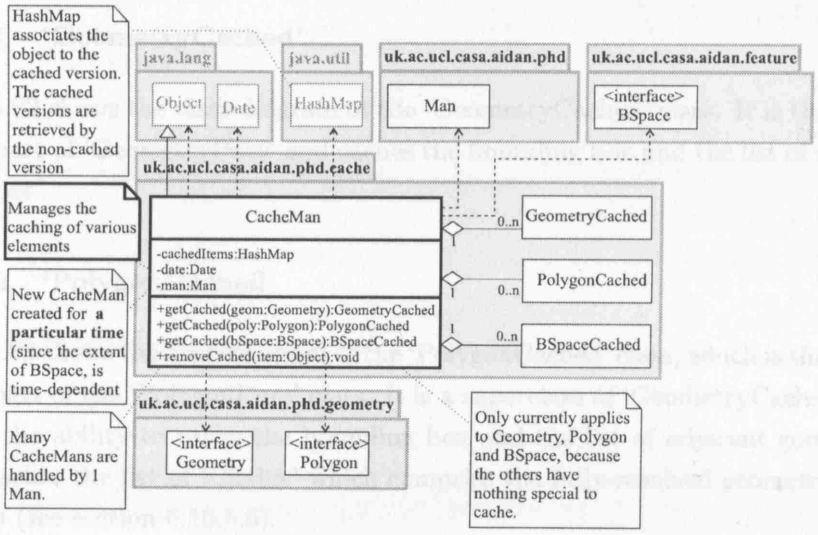


Figure 6.81: UML class diagram of 'CacheMan', showing its fields, methods and dependencies.

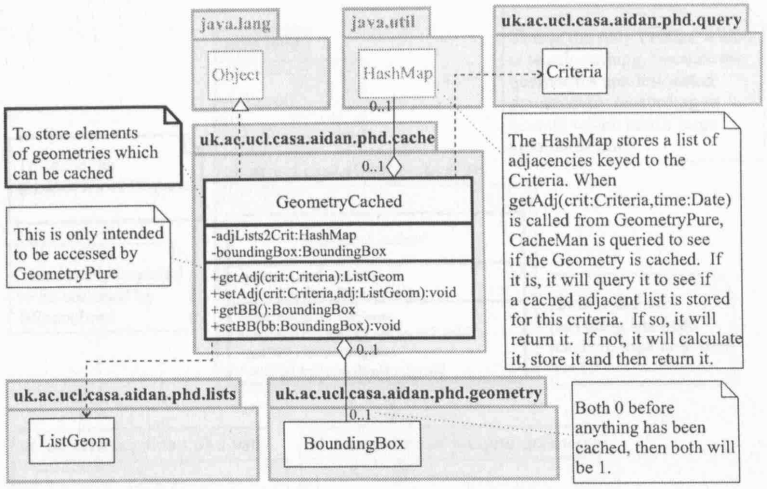


Figure 6.82: UML class diagram of 'GeometryCached', showing its fields, methods and dependencies.

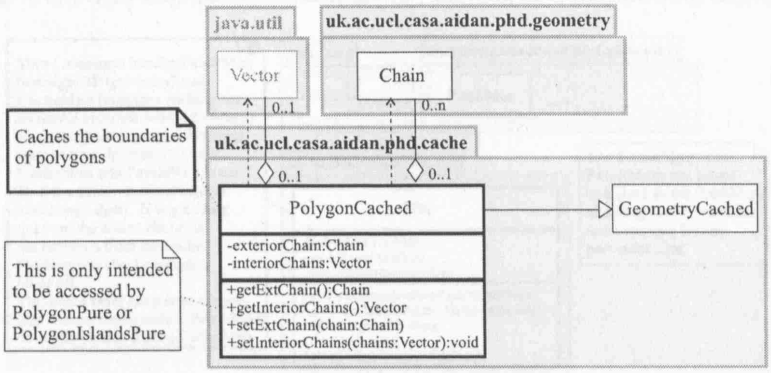


Figure 6.83: UML class diagram of 'PolygonCached', showing its fields, methods and dependencies.

6.15.2 Cacheable objects

6.15.2.1 'GeometryCached'

Figure 6.82 shows the class diagram of the 'GeometryCached' class. It is the cached counterpart of 'GeometryPure' and caches the bounding box and the list of adjacent geometries.

6.15.2.2 'PolygonCached'

Figure 6.83 shows the class diagram of the 'PolygonCached' class, which is the cached counterpart of the 'PolygonPure' class. It is a superclass of 'GeometryCached', thus inherits the ability to cache the bounding box and the list of adjacent geometries. It also caches the list of 'Chains' which comprise the fully-resolved geometry of the polygons (see section 6.10.5.5).

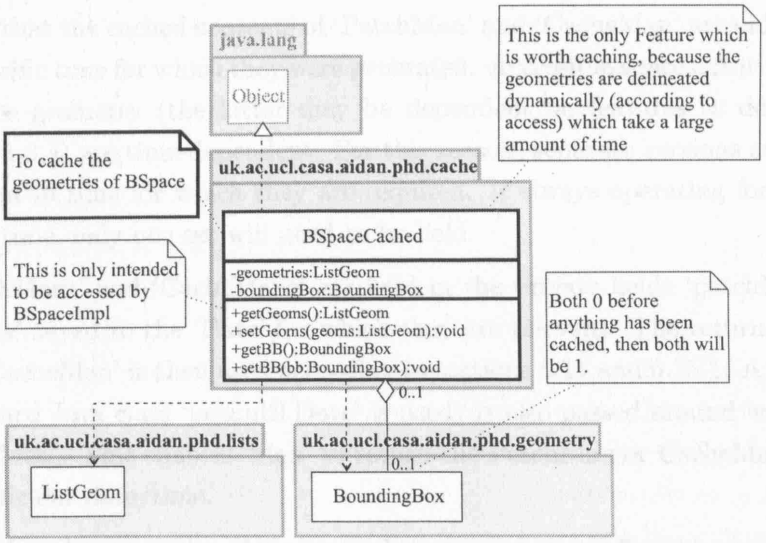


Figure 6.84: UML class diagram of 'BSpaceCached', showing its fields, methods and dependencies.

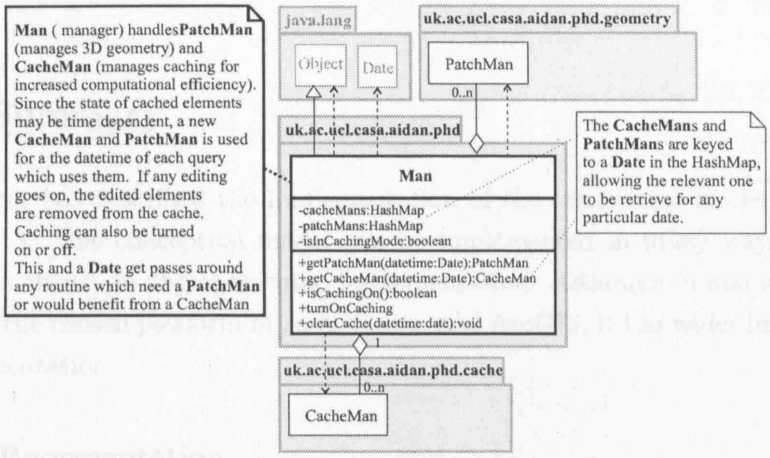


Figure 6.85: UML class diagram of 'Man', showing its fields, methods and dependencies.

6.15.2.3 'BSpaceCached'

Figure 6.84 shows the class diagram of the 'BSpaceCached' class, whose cached counterpart is 'BSpaceImpl'. This is the only feature (in the 'FeatureImpl' hierarchy) which is cacheable. As can be seen from figure 6.84, it is the result of the 'getBB' and 'getGeoms' methods that are cached. The reason that the results of 'getGeoms' is cached is because it is a time- and pedestrian-dependent query which takes some time to compute.

6.15.3 'Man'

Figure 6.85 shows the class diagram of the 'Man' class. One instance of the 'Man' class is passed around all the functions which require access to or all the functions which are dependent on functions requiring access to any 'PatchMan' or 'CacheMan'.

It is likely that the cached contents of 'PatchMan' and 'CacheMan' are only relevant for the specific time for which they were generated. Accessible spaces, feature extents and surface geometry (the latter may be dependent on features as described in section 6.11.3.4) are time-dependent. For this reason, separate versions are held for each instant in time for which they are required. If always operating for the same instant in time, only one set will need to be held.

The 'PatchMans' and 'CacheMans' are held in the private fields 'patchMans' and 'cacheMans' keyed to the 'Date' for which they are relevant. The returned 'PatchMan' or 'CacheMan' is then used according to sections 6.11 and 6.15.1. A date/time (the standard Java class 'java.util.Date' is used) is also passed around with the instance of 'Man'. This enables 'Man' to return the PatchMan or CacheMan relevant to the particular date/time.

'Man' also has the capability of having caching turned on or off and having the cache cleared. This is useful for operations which may result in a change of state of cached objects.

6.16 Summary

This chapter has described the implementation of the conceptual model presented in chapter 5. The conceptual model can be implemented in many ways and this chapter has shown an object-oriented implementation. Although in many ways it is specific to the chosen platform of Java, Ozone and ArcGIS, it has wider implications for implementation.

6.16.1 Representation

The data model chosen is designed to be as close to the entities of the conceptual model as possible; thus it illustrates how the entities can fit together in an implementation. It uses an object-oriented methodology for formalising concepts. These are illustrated in standard UML diagrams in a way which communicates the immediate relationships to other concepts. As discussed, this approach is useful for considering any system. Even if relational database technology is used for storage, the way in which the system is constructed is useful.

A large number of supporting classes have been defined. Many of these have been defined in order to package concepts into smaller classes in order for them to be used in more than one part of the implementation. Examples are the 'Criteria', 'Restrictions', 'TimeStamps', and the 'Lists' with built-in time-stamping. The further formalisation of concepts and the identification of shared concepts is essential for building an implementation.

However, clearly some of the differences between the conceptual and logical models are due to specific implementation-dependent factors. Most notably, differences

arise due to Ozone's requirement for its database objects to implement an interface and have proxy classes defined. Within this, the most significant implementation problem was with the geometries, resulting in the complexity around the design of the classes which implement the 'Geometry' interface. This illustrates that there are indeed implementation-specific considerations.

In order to conform to one of the original guidelines of the model design, the data volume has been kept down. In the implementation, a small amount of 'redundancy' has been introduced for performance reasons, but this has been kept to a minimum. For example, in the conceptual model, the 'Feature' entities have a list of the 'Geometry' entities which comprise their geometrical extent but the 'Geometry' entities do not know which features' extents' they comprise. In the implementation, instances of both 'GeometryImpl' and 'FeatureImpl' maintain lists of references to each other. If the instances of 'GeometryImpl' did not have a list of the 'Features' they comprise, at worst all the instances of 'GeometryImpl' would have to be searched and at best there would be some kind of indexing facility or a caching facility in which the result of the first search would be stored and used subsequently.

Caching is also used by the implementation which is handled by the 'CacheMan' class (section 6.15.1), but this is only held temporarily (in the implementation, this is held in RAM, which causes some memory problem for large areas). Since cached values may only be valid for a particular time, 'CacheMan' also handles different versions of cached objects. The way in which caching is handled can be applied more widely to other implementations.

6.16.2 Algorithms

The way in which the 'PatchMan' and 'Patch' classes work illustrates how the '3DReasoner' entity can be implemented. Thus the operation of 'PatchMan' and 'Patch' classes has wider implications outside this specific implementation.

6.16.3 Data input and output

In order to populate any data model the facility to input data is essential. The creation of a streamlined editing environment for the input of data lay beyond the scope of this thesis. However, it is clear that for development and evaluation, the model needed to be populated. Thus, the solution proposed is rather clumsy but it highlights what type of data are required by the model. The data are input as a directory full of Shapefiles and DBF tables which contain the geometry and topology required by the model and references to each other. The data are prepared using some manual editing and VBA scripts which automatically perform a variety of operations using ArcGIS's 2D topology tools. Since the data required are well-defined, a more streamlined implementation of an editing and data-preparation environment can be built.

Since the implementation is entirely console (command line) based, spatial data are output as shapefiles for inspection in ArcGIS. An implementation can just as easily be output on screen or in another output file format.

6.16.4 Proof of concept

The implementation was an important part of the development of the model and for its evaluation. The fact that the implementation achieved its goal is capable of modelling the situations it was designed validates the conceptual model design.

Part III

Evaluation

Chapter 7

Evaluation

Chapter 2 reviewed some of the types of potential applications which are in scope. Chapter 3 identified a list of design principles to be adhered to in the light of the application review. Chapter 5 presented a conceptual model which fulfills these design principles and then chapter 6 presented a proof-of-concept implementation which included some wider implementation issues.

This chapter with evaluate the model. Firstly, it will evaluate it in terms of the representation ability using nine case studies (section 7.2). Secondly, it will evaluate the model from a technical and computational perspective (sections 7.3, 7.4 and 7.5).

7.1 Scope of the conceptual framework

The scope of the conceptual model was to provide a description of the 3D structure of functional spaces in the environment (particularly the urban environment). The following set of design principles were defined and explained in chapter 3:

- the ability to describe different conceptualisations of features
- the seamless treatment of space exterior and interior to buildings
- pedestrian accessibility
- the concept of a data repository to which information can be added, in an incremental fashion
- the storage of 2D and 3D geometry.
- the representation of time

7.2 Worked examples

This chapter will present some simple worked examples. These examples are not applications or end-goals in themselves; rather they are little examples which were used

in the development of the model, which can be used to illustrate representational ability.

As described in section 6.6.1, data are entered into the prototype through three shapefiles (with attributes) and three figures for each layer and seven additional figures describing the features. It does not matter how the information is split into layers, but each layer must have a planar topology (2-manifold), must not self-overlap in 2D and must have an explicit 2D topology built. Each layer in each example is numbered from zero, and, as will be seen, these index values are used to refer between layers for the 'RefGeom' and 'Topology' fields. These files contain a mixture of the raw data and the automatically-built and derived data from these raw data. The raw data themselves are all the line primitives, some of the node primitives, a subset of the attributes of the nodes, lines and polygons and the figures describing the features. Derived data are most of the nodes and all of the polygons which are automatically built from the lines. Some of these derived geometrical primitives have user-defined attributes added, e.g. polygon 4 in figure 7.4 is marked as being a ramp.

In these worked examples, the details of how the raw input data are provided is not important; rather it is *what raw data are provided.* The following case studies show what data were originally input and the actual output results of the '3DReasoner' and the 'AccessResolver' entities (sections 5.4 and 5.7) from the prototype. These illustrate the results of the implementation of the conceptual model.

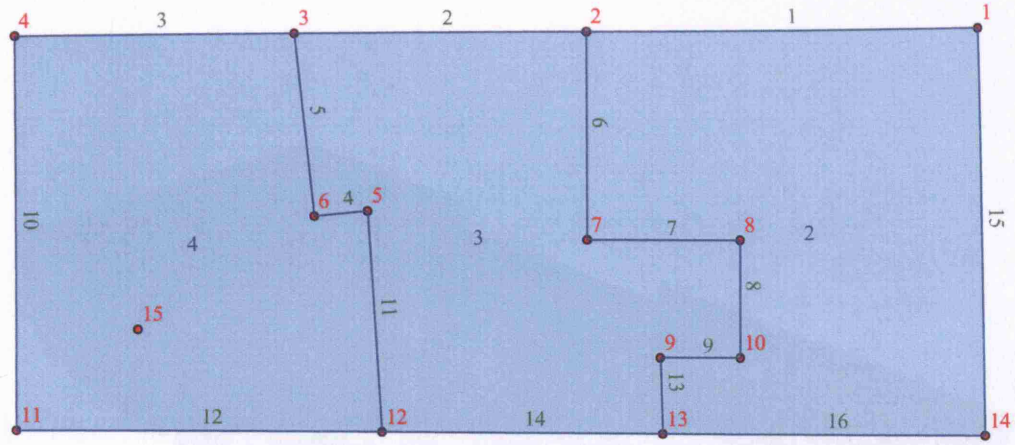
7.2.1 Case study 1 – simple split levels

This very simple example demonstrates how a situation in which there are three small horizontal areas on three different levels at three heights all surrounded by a wall can be described by a small set of input data. The user-provided geometrical and topological information is shown in figure 7.1; the user-defined feature information is shown in figure 7.2.

7.2.1.1 Geometry

Figures 7.1 and 7.2 show the raw data component of the input files for producing the output in figure 7.3. Of the geometries shown, 16 lines (all) and one node (node 15) were provided by the data provider; the others were built automatically. All the attribution shown was provided (although those which are of the most basic type can be assumed to be 'Node', 'Line' and 'Polygon' by default). Additionally one feature has been described.

All this is input as one layer (but could be input as multiple layers). There are three polygons in total which are separated lines representing an unspecified offset. Each of these has a node on (nodes 5 and 7) with a height specified which is relative to



ID	Type	Ht/Off	RefGeom
1	Node		
2	Node		
3	Node		
4	Node		
5	NodeOff	-2	p1_4
6	Node		
7	NodeOff	1.5	p1_3
8	Node		
9	Node		
10	Node		
11	Node		
12	Node		
13	Node		
14	Node		
15	NodePhys	35	

ID	Type	Features
1	Line	wall1
2	Line	wall1
3	Line	wall1
4	LineOff	
5	LineOff	
6	LineOff	
7	LineOff	
8	LineOff	
9	LineOff	
10	Line	wall1
11	LineOff	
12	Line	wall1
13	LineOff	
14	Line	
15	Line	wall1
16	Line	wall1

ID	Type	Features
2	Polygon	space1
3	Polygon	space1,bspace1
4	Polygon	space1

Figure 7.1: All the user-defined geometrical data and associated attribute data entered for case study 1 is shown as nodes (red, left figure), lines (green, middle figure) and polygons (blue, right figure). Fields which are empty are omitted. The 'ID' field refers to the corresponding number of the geometry in the figure; the 'Type' field gives the type of geometrical primitive; the 'Ht/Off' field gives the absolute or relative height (depending on type); the 'RefGeom' field is the relative geometry field which applies to geometries with a relative height attached (where 'n', 'l' and 'p' indicate the geometry type, the subsequent number refers to the input layer number and the number after the underscore refers to the ID of the geometrical primitive); and 'Feature' is a list of all the features for which the geometry applies. In this example, the only geometrical primitives with height values are two 'NodeOffs' and a 'NodePhys'. The 3D output generated by the prototype is shown in figure 7.3.

'Wall' Features			
Name	**Strength**	**ht**	**Cyclic existence**
wall1	secure	10	

'Space' Features		
Name	**Access Restrictions**	**Cyclic existence**
space1		

'BoundedSpace' Features			
Name	**Access Restrictions**	**Cyclic existence**	**Pedestrian**
bspace1			maxStep=30

Figure 7.2: The features in case study 1

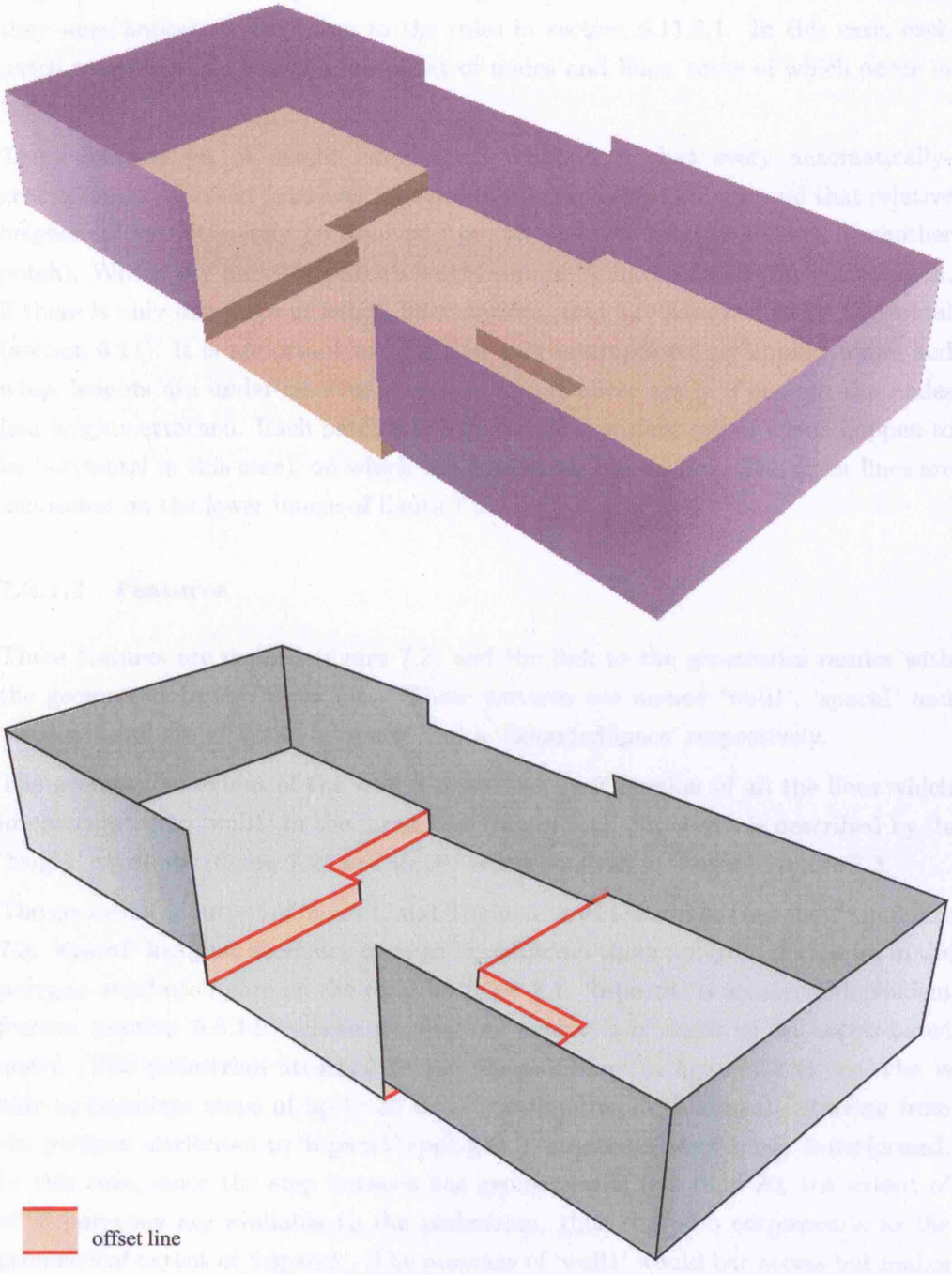


Figure 7.3: 3D output of case study 1 by the '3D Reasoner' using the geometrical information shown in figure 7.1.

one of the adjacent polygons. *One* absolute height exists in polygon 4, from which all the other heights are 'reasoned'.

This example is similar to those illustrated in figures 5.8 and 5.9 (pages 159 and 160). The '3DReasoner' splits this set of geometries into 3 patches (no matter how they were imported) according to the rules in section 6.11.3.1. In this case, each patch comprises one polygon and a set of nodes and lines, some of which occur in two patches.

The minimum set of height information required is that every automatically-generated patch has at least one absolute or relative height datum, and that relative heights can be ultimately be resolved from an absolute height (perhaps in another patch). Where any part of a patch's height cannot be interpolated (the entire patch, if there is only one piece of height information), they are assumed to be horizontal (section 6.11). It is important to note that this assumption only applies where and when heights are under-resolved; the rule would never apply if enough the nodes had heights attached. Each patch has a geometrical surface (all of which happen to be horizontal in this case), on which the geometries are draped. The offset lines are annotated on the lower image of figure 7.3.

7.2.1.2 Features

Three features are defined (figure 7.2) and the link to the geometries resides with the geometries in the input files. These features are named 'wall1', 'space1' and 'bspace1' and are a 'Wall', a 'Space' and a 'BoundedSpace' respectively.

The geometrical extent of the wall is described by the union of all the lines which are attributed to 'wall1' in the input files (figure 7.1). Its height is described by its 'height' attribute (figure 7.2) and its 3D representation is shown in figure 7.3.

The geometrical output of 'space1' and 'bspace1' are identical to that shown in figure 7.3. 'space1' has that geometry because it comprises three polygons, as shown in the polygon attribute figure on the right in figure 7.1. 'bspace1' is an access-dependent feature (section 5.5.1.2) whose geometrical extent is a result of an access-based query. The pedestrian attached to the 'BoundSpace' in figure 7.2 is one who is able to negotiate steps of up to 30 units (centimetres in this case). Starting from the polygon attributed to 'bspace1' (polygon 3) an access-based query is performed. In this case, since the step between the geometries is less than 30, the extent of all 3 polygons are available to the pedestrian, thus this also corresponds to the geometrical extent of 'bspace1'. The presence of 'wall1' would bar access but makes no difference in this case because this is at the edge of the area.

7.2.1.3 Discussion

This very simple example shows how a situation which is common in the urban environment can be described with just a small set of input data. These input data

are a set of 2D geometrical primitives with a small amount of height and surface morphology information and some simple features.

7.2.2 Case study 2 – ramp and bridge

The user-provided geometrical and topological information for this case study is shown in figures 7.4 and 7.5. The user-defined feature information is shown in figure 7.6. For this case study, user-defined information were:

- the geometry of 55 lines (with the remainder being built automatically)
- all the attributes shown
- one feature

This is also a simple example, but it complicated by the fact that there is more than one overlapping layer. It needs to be input as two separate layers and there needs to be information about how the layers connect to each other. These are shown in figures 7.4 and 7.5. 'space2' is described in figure 7.6.

7.2.2.1 Geometry

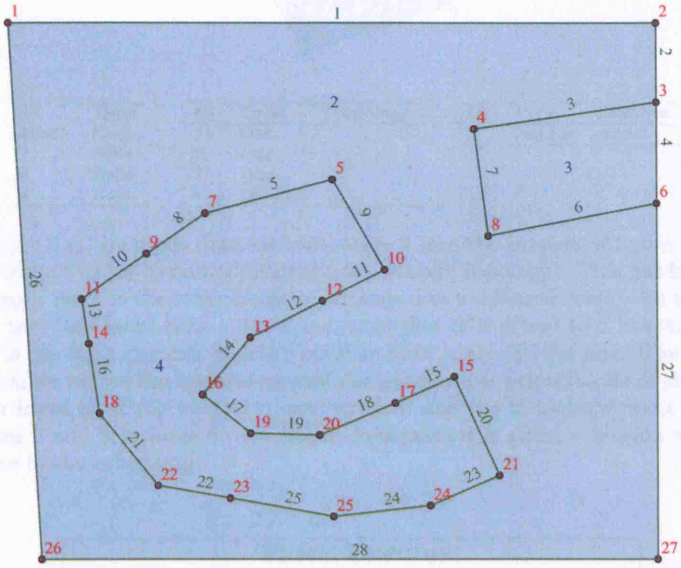
The 3D output produced by the '3DReasoner' entity from the two layers of raw data (figures 7.4 and 7.5) is shown in figure 7.7. The 'Topology' field in the polygon attribute figure shown in figure 7.5 has resulted in the two layers being topologically connected. When lines topologically join they are automatically treated as breaklines.

As in the last example, the minimum set of heights is given, such that every patch has height information. This example has four patches. Heights for each patch can be assigned thus:

- The patch corresponding to polygon 2 in layer 1 takes a height from the absolute height assigned to node 2 in layer 1
- The patch corresponding to polygon 3 in layer 1 takes a height from the relative height assigned to node 3 which is relative to node 2 in layer 1.
- The patch corresponding to polygon 2 in layer 2 takes a height from the patch corresponding to polygon 3 in layer 1, by virtue of the fact that line 7 in layer 1 is a breakline.
- The patch corresponding to polygon 4 in layer 1 takes heights from the two patches connected by the breaklines.

This process is described in section 6.11 and is illustrated in the UML sequence diagram in figure 6.55.

The patch corresponding to polygon 4 in layer 1 is a 'ramp', so rather than being treated as horizontal, its surface geometry is calculated according to figure 6.59. It must be clear where the two entry points are for polygons marked as ramps or stairs.

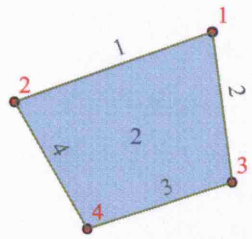


ID	Type	Ht/Off	RefGeom
1	Node		
2	NodePhys	2	
3	NodeOff	2.5	p1_2
4	Node		
5	Node		
6	Node		
7	Node		
8	Node		
9	Node		
10	Node		
11	Node		
12	Node		
13	Node		
14	Node		
15	Node		
16	Node		
17	Node		
18	Node		
19	Node		
20	Node		
21	Node		
22	Node		
23	Node		
24	Node		
25	Node		
26	Node		
27	Node		

ID	Type	Ht/Off
1	Line	
2	Line	
3	LineOff	
4	Line	
5	LineOff	
6	LineOff	
7	LineOff	
8	LineOff	
9	LineOff	
10	LineOff	
11	LineOff	
12	LineOff	
13	LineOff	
14	LineOff	
15	LineOff	
16	LineOff	
17	LineOff	
18	LineOff	
19	LineOff	
20	LineOffValue	0
21	LineOff	
22	LineOff	
23	LineOff	
24	LineOff	
25	LineOff	
26	Line	
27	Line	
28	Line	

ID	Type	Subtype Features	
2	Polygon		space2
3	Polygon		space2
4	Polygon	ramp	space2

Figure 7.4: Layer 1 of the input data for case study 2 (see the caption of figure 7.1 for notation). In this example, polygon 4 has been marked as a ramp and is surrounded by offset lines. There is a break of slope at the base of the ramp (line 20) represented at an offset line with a relative height of '0' (see figure 6.3). Node 2 has an absolute height and node 3 has a relative height which gives a bridge height. 'space8' is the 'Space' feature (described in figure 7.6). The actual bridge itself it defined as another input layer (because of the need for input layers to have a planar topology). This is shown in figure 7.5.



ID	Type
hline 1	Node
2	Node
3	Node
4	Node

ID	Type	Topology
1	Line	
2	Line	l1_7
3	Line	
4	Line	l1_9

ID	Type	Features
2	Polygon	space2

Figure 7.5: Layer 2 of the input data for case study 2 (see the caption of figure 7.1 for notation). The polygon attribute figure has an additional field named 'Topology'. This has information about how this input layer links to the other input layer (shown at a different scale). Its value works in the same way as in the 'RefGeom' field – the 'l' indicates that it is linked to a line, the number before the underscore is the layer number and the number after is the 'ID' of line. The 'Topology' value has the layer number before the underscore and the geometrical primitive ID of the same geometry type. The other input layer (figure 7.4) is numbered '1' and the 'Topology' value links this section of bridge to lines 7 and 9 in layer 1. No height information is given – heights will eventually be taken from those in the other layer.

'Space' Features	
Name	**Access restrictions Cyclic existence**
space2	

Figure 7.6: The feature in case study 2

As shown in figure 6.57, this is done using two breaklines. The centreline of access is generated followed by the 3D geometry. The way this is done is shown in figure 6.59.

Although this sounds a complicated procedure, this can be managed using the 'PatchMan' and 'Patch' classes described in section 6.11. Again, it is emphasised that the example is using minimal height data and that under-resolved height data are used.

7.2.2.2 Features

The only feature in this example is 'space2' whose geometrical extent is the union of the extents of all the polygons.

7.2.2.3 Discussion

Two of the most novel aspects of the model are:

- surface morphology information – the offset line
- height constraints in specific and significant places

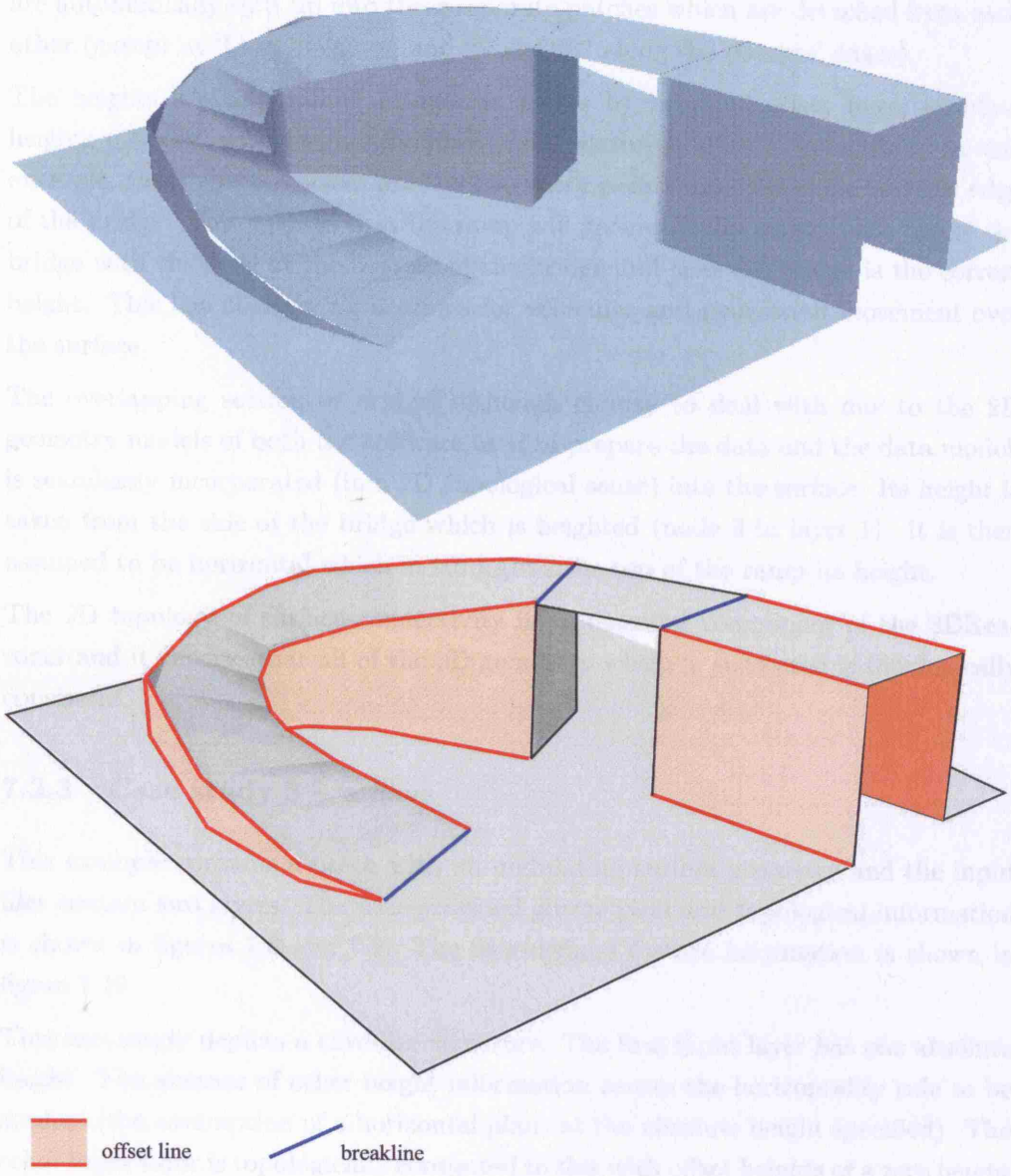


Figure 7.7: 3D output of case study 2.

These two types of information are used to construct 'patches' which formalise the way in which these can be used to generate a surface geometry.

In this example, there is no space under the ramp or under the edges of the bridge, so apart from the overlapping section, all this has been imported as one layer (they could be imported as more than one layer). The ramp and bridge are surrounded by offset lines (and breaklines which are a special case of an offset line), so these are automatically split up into three separate patches which are detached from each other (except at 'LineOffValues' and 'NodeOffs' along the patches' edges).

The heights are constrained in specific places by heighted offset lines, absolute heights ('Nodes' with heights assigned) and relative heights ('NodeOffs'). In this example, these specific places are the two entry points onto the slope and the edge of the bridge. This ensures that the ramp will geometrically connect the top of the bridge with the level at the bottom of the bridge and that the bridge is the correct height. This has obvious implications for vehicular and pedestrian movement over the surface.

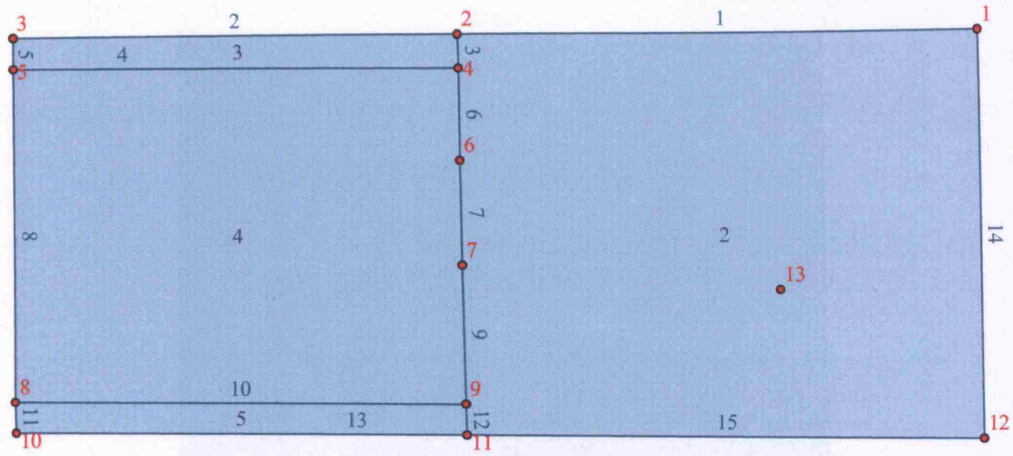
The overlapping section of bridge, although clumsy to deal with due to the 2D geometry models of both the software used to prepare the data and the data model, is seamlessly incorporated (in a 2D topological sense) into the surface. Its height is taken from the side of the bridge which is heighted (node 3 in layer 1). It is then assumed to be horizontal which in turn gives the top of the ramp its height.

The 2D topology of surface connectivity is an essential component of the 3DReasoner and it ensures that all of the 3D geometry which is generated is topologically consistent.

7.2.3 Case study 3 – a cave

This example contains a patch with an undulating surface geometry and the input files contain two layers. The user-provided geometrical and topological information is shown in figures 7.8 and 7.9. The user-defined feature information is shown in figure 7.10.

This case study depicts a cave-like structure. The first input layer has one absolute height. The absence of other height information causes the horizontality rule to be applied (the assumption of a horizontal plane at the absolute height specified). The other input layer is topologically connected to this with offset heights of a zero height (i.e. breaklines) with respect to the first layer. Since the heights are with respect to the first layer, they do not affect the heights on this layer; instead they cause the second input layer to be 'pulled down' at the edges to meet it. The '3DReasoner' takes interpolated heights from the patch corresponding to the first input layer and assigns these to the edges of the patch corresponding to the second input layer.



ID	Type	Ht/Off
1	Node	
2	Node	
3	Node	
4	Node	
5	Node	
6	Node	
7	Node	
8	Node	
9	Node	
10	Node	
11	Node	
12	Node	
13	NodePhys	23

ID	Type	Ht/Off	Features
1	Line		
2	Line		
3	Line		wall2
4	Line		
5	Line		
6	Line		wall2
7	Line		
8	Line		
9	Line		wall2
10	Line		
11	Line		
12	Line		wall2
13	Line		
14	Line		
15	Line		

ID	Type	Features
2	Polygon	space3
3	Polygon	space3
4	Polygon	space3
5	Polygon	space3

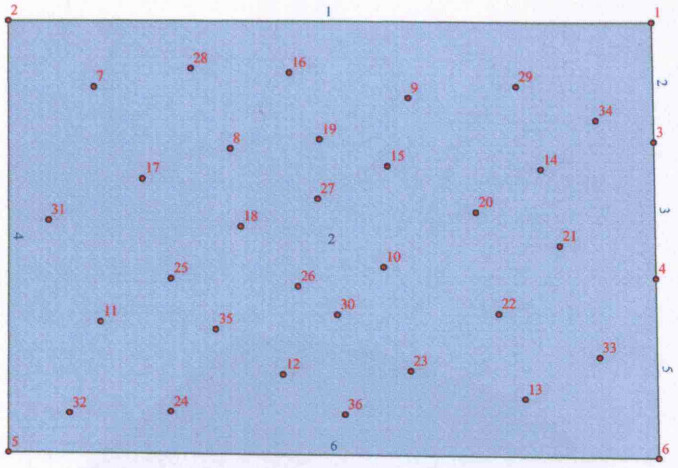
Figure 7.8: Layer 1 of the input data for case study 3 (see the caption of figure 7.1 for notation). In this layer, there only one absolute height and there is one 'Wall' feature and one 'Space' feature. Some of the lines exist purely for attachment to other geometries in the other layer (e.g. lines 3 and 10) in figure 7.9.

7.2.3.1 Discussion

This example illustrates that it is possible to specify undulating surfaces (most of the examples presented here do not specify undulating surfaces). The upper surface of the cave (specified in the second input layer) is topologically linked to the layer below. The reference geometry on the offset line at the edge of the second layer, is a geometry on the first input layer; this ensures that the upper surface does not play a part in determining the surface geometry of the lower layer. The lower layer's surface geometry is resolved first, and then the reasoned heights are assigned to the edge of the upper layer. As with the previous case study, it shows the dependence of the 3D geometry on the surface connectivity topology.

7.2.4 Case study 4 – a building with two split-level storeys

This example considers two storeys of a building with split levels. It is input in three layers. The user-provided geometrical and topological information is shown in



ID	Type	Ht/Off	RefGeom
1	Node		
2	Node		
3	Node		
4	Node		
5	Node		
6	Node		
7	NodeOff	2	p1_4
8	NodeOff	4.6	p1_4
9	NodeOff	2	p1_4
10	NodeOff	8.5	p1_4
11	NodeOff	2	p1_4
12	NodeOff	3.3	p1_4
13	NodeOff	2.4	p1_4
14	NodeOff	5	p1_4
15	NodeOff	5.5	p1_4
16	NodeOff	1.3	p1_4
17	NodeOff	3.7	p1_4
18	NodeOff	6.5	p1_4
19	NodeOff	4.9	p1_4
20	NodeOff	7.5	p1_4
21	NodeOff	8	p1_4
22	NodeOff	7.5	p0_4
23	NodeOff	4.5	p1_4
24	NodeOff	1	p1_4
25	NodeOff	4	p1_4
26	NodeOff	6	p1_4
27	NodeOff	7	p1_4
28	NodeOff	1.4	p1_4
29	NodeOff	1.9	p1_4
30	NodeOff	7	p1_4
31	NodeOff	0.9	p1_4
32	NodeOff	1.3	p1_4
33	NodeOff	3	p1_4
34	NodeOff	3	p1_4
35	NodeOff	3	p1_4
36	NodeOff	1.5	p1_4

ID	Type	Ht/Off	RefGeom	Topology
1	LineOffValue	0	p1_4	l1_4
2	Line			
3	Line			
4	LineOffValue	0	p1_4	l1_8
5	Line			
6	LineOffValue	0	p0_4	l10_10

ID	Type	Features
2	Polygon	space3

Figure 7.9: Layer 2 of the input data for case study 3 (see the caption of figure 7.1 for notation). This single polygon has the same 2D geometry as polygon 4 in the other layer in figure 7.8 (these two layers are shown at different scales) and has a set of nodes with relative heights given with respect to polygon 4 in the other layer. The edges of this polygon are topologically linked to the layer below.

'Space' Features		
Name	**Access restrictions**	**Cyclic existence**
space3		

'Wall' Features			
Name	**Strength**	**ht**	**Cyclic existence**
wall2	secure	6	

Figure 7.10: The features in case study 3

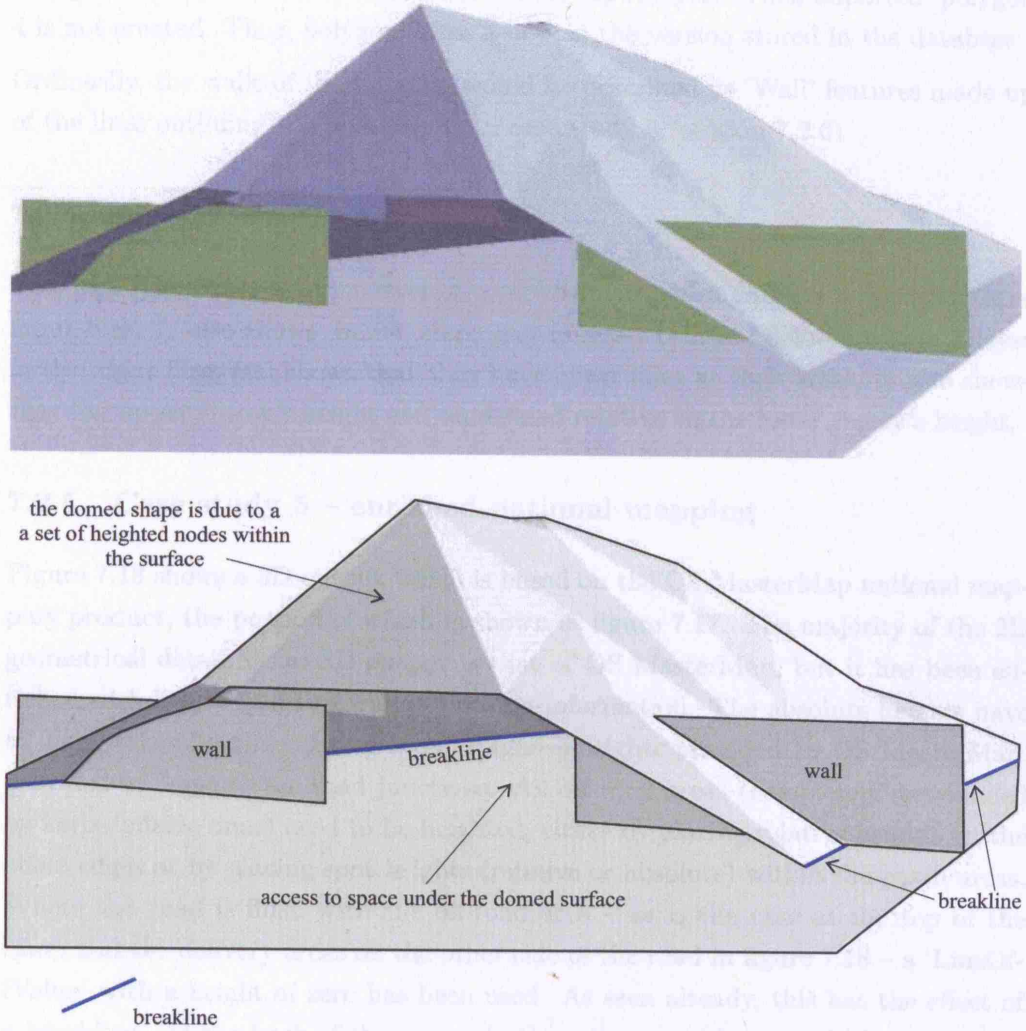


Figure 7.11: 3D output of case study 3.

figures 7.12, 7.13 and 7.14; the user-defined feature information is shown in figure 7.15.

The 3D output of the case study in figure 7.16 is of the 'bspace2' feature which represents the connected space from polygon 3 in layer 1 for the pedestrian attached to the feature definition (figure 7.15). Since the pedestrian (which is defined in the definition of the 'bspace2' feature as "maxStep=30") is able to negotiate all the steps and ramps, the whole area is accessible.

Polygon 4 in layer 2 is marked as a hole in the input layer. When imported, polygon 4 is not created. Thus, polygon 2 has a hole in the version stored in the database.

Ordinarily, the walls of the building would be described as 'Wall' features made up of the lines outlining the building, as in case study 6 (section 7.2.6).

7.2.4.1 Discussion

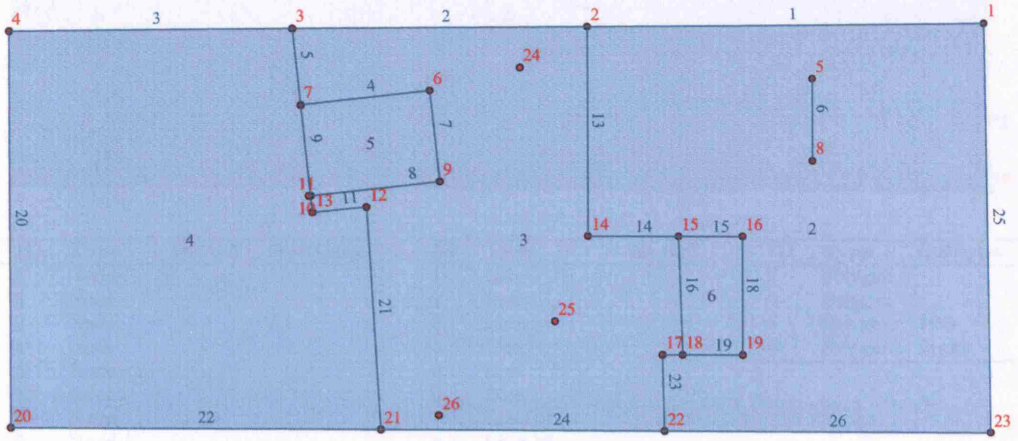
This example shows another level of complexity in which there is a need for three input files. It also shows 'inline' steps and ramps which are defined within a layer in the input files, and shows that they have offset lines at their sides. It also shows that the upper storey's height can be defined relative to the lower storey's height.

7.2.5 Case study 5 – enriched national mapping

Figure 7.18 shows a 3D output which is based on the OS MasterMap national mapping product, the portion of which is shown in figure 7.17. The majority of the 2D geometrical detail in the 3D output is that of OS MasterMap, but it has been enriched with height and surface morphology information. The absolute heights have all been taken by the existing – but sparse – heights provided by OS MasterMap, provided at some of the road junctions. All off-road areas (completely surrounded by kerbs/offside lines) need to be heighted, either by placing relative heights on the offset edges or by placing spot heights (relative or absolute) within the patch areas. Where the road is flush with the off-road area – as is the case at the top of the ramp and the delivery areas on the other side of the road in figure 7.18 – a 'LineOffValue' with a height of zero has been used. As seen already, this has the effect of a breakline. At the back of the car park, there is a curved ramp which leads to an underground car park (the underground car park is not shown in the figure). With a small amount of height and surface morphology information, a 3D geometry can be generated. The offset lines have the effect detaching the height and orientation of the surfaces from those which they separate.

7.2.5.1 Discussion

The fact that this example is based on national mapping data with a small amount of additional information, shows how the mapping concepts of OS MasterMap are

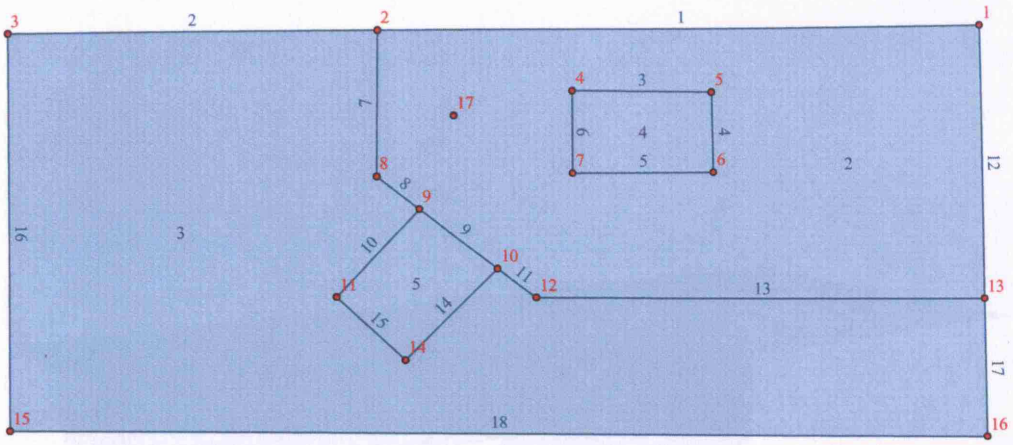


ID	Type	Ht/Off	RefGeom
1	Node		
2	Node		
3	Node		
4	Node		
5	Node		
6	Node		
7	Node		
8	Node		
9	Node		
10	Node		
11	Node		
12	NodeOff	6	p1_3
13	Node		
14	NodeOff	3	p1_3
15	Node		
16	Node		
17	Node		
18	Node		
19	Node		
20	Node		
21	Node		
22	Node		
23	Node		
24	Node		
25	Node		
26	NodePhys	45	

ID	Type	Ht/Off
1	Line	
2	Line	
3	Line	
4	LineOff	
5	LineOff	
6	Line	
7	LineOffValue	0
8	LineOff	
9	LineOffValue	0
10	Line	
11	LineOff	
12	LineOff	
13	LineOff	
14	LineOff	
15	LineOff	
16	LineOffValue	0
17	LineOff	
18	LineOffValue	0
19	LineOff	
20	Line	
21	LineOff	
22	Line	
23	LineOff	
24	Line	
25	Line	
26	Line	

ID	Type	SubType	Features
2	Polygon		
3	Polygon		bspace2
4	Polygon		
5	Polygon	Stairs	
6	Polygon	Ramp	

Figure 7.12: Layer 1 of the input data for case study 4 (see the caption of figure 7.1 for notation). This depicts a split level lower storey of a building, access to components of the split level being provided by some steps and a ramp. Line 6 exists in order for the main stairs in the third layer (layer 2, figure 7.8) to topologically join. There is one absolute height in the middle portion of the split levels. The other portions' heights are relative to the height of the middle portion. One of the portions is part of the 'BoundedSpace' ('bspace2').

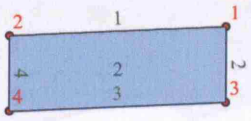


ID	Type	Ht/Off	RefGeom
1	Node		
2	Node		
3	Node		
4	Node		
5	Node		
6	Node		
7	Node		
8	Node		
9	Node		
10	Node		
11	Node		
12	NodeOff	6	p2_3
13	Node		
14	Node		
15	Node		
16	Node		
17	NodeOff	15	p1_2

ID	Type	Ht/Off
1	Line	
2	Line	
3	Line	
4	Line	
5	Line	
6	Line	
7	LineOff	
8	LineOff	
9	LineOffValue	0
10	LineOff	
11	LineOff	
12	Line	
13	LineOff	
14	LineOff	
15	LineOffValue	0
16	Line	
17	Line	
18	Line	

ID	Type	Subtype
2	Polygon	
3	Polygon	
4	Polygon	Hole
5	Polygon	Stairs

Figure 7.13: Layer 2 of the input data for case study 4 (see the caption of figure 7.1 for notation). This depicts a split-level upper storey of a building where the portions are connected by steps. There is a hole (a polygon is marked as such in the input files, which causes an absence of polygon in the layer as imported) which is where the staircase will emerge, its base being connected to line 6 in layer 1. The height of this layer is given with respect to the lower storey (node 17) and the separation between the two split-level portions is provided by node 12 which is along the offset line.



ID	Type
1	Node
2	Node
3	Node
4	Node

ID	Type	Ht/Off	Topology
1	Line		
2	LineOffValue	0	l1_6
3	Line		
4	LineOffValue	0	l3_6

ID	Type	SubType
2	Polygon	Stairs

Figure 7.14: Layer 3 of the input data for case study 4 (see the caption of figure 7.1 for notation). This layer depicts the staircase which connects layers 1 and 2. Information about the connection is contained in the 'Topology' field of the attribute figure of the lines.

'BoundedSpace' Features			
Name	**Access Restrictions**	**Cyclic existence**	**Pedestrian**
bspace2			maxStep=30

Figure 7.15: The feature in case study 4

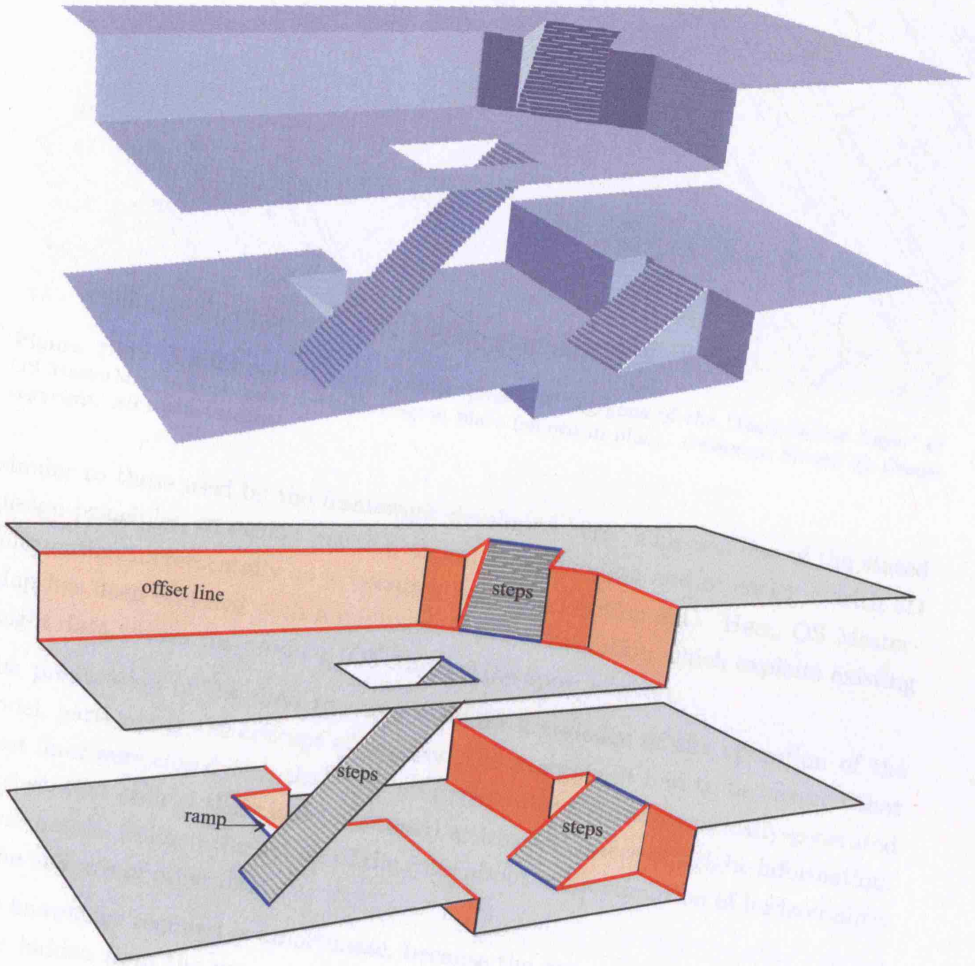


Figure 7.16: 3D output of case study 4 (seen from the other side with respect to the input files in figures 7.12, 7.13 and 7.14).

Figure 7.17: A small portion of the points, lines and polygons of the 'Topographic Layer' of OS MasterMap incorporating 1-19 Torrington place (shown in blue). *Ordnance Survey © Crown copyright. All rights reserved.*

similar to those used by the framework developed here. This was one of the stated design principles, to exploit existing digital mapping data and to enrich it with 3D information incrementally as it becomes available (section 3.4). Here, OS MasterMap has been enriched with a minimum set of information which exploits existing height data within the product (OS MasterMap spot heights).

The preparation of the data required a sound knowledge of the operation of the model, particularly the concept of patches. For example, it had to be ensured that offset lines were closed such that they properly defined the automatically-generated patches, that each of these had (or shared) at least one piece of height information. It was helpful to know the details of the rules about the assumption of horizontality, in the absence of other data.

This knowledge required is unfortunate, because the concept of patches is intended to be hidden from the user of the framework; indeed, the 'Patch' class has been implemented as a private class visible only to the 'PatchMan' class which deals with all the heighting transparently. This is not necessarily a fundamental problem, though. Just as patches are built automatically, it is entirely possible to build a system which can identify when height data are under-resolved and advise the data user which areas are lacking in data definition. Similarly, the assumptions of the '3DReasoner' could be further developed. For example, in the absence of other heighting and orientation data, patches could be heighted and oriented according to the height and orientation of surrounding patches.

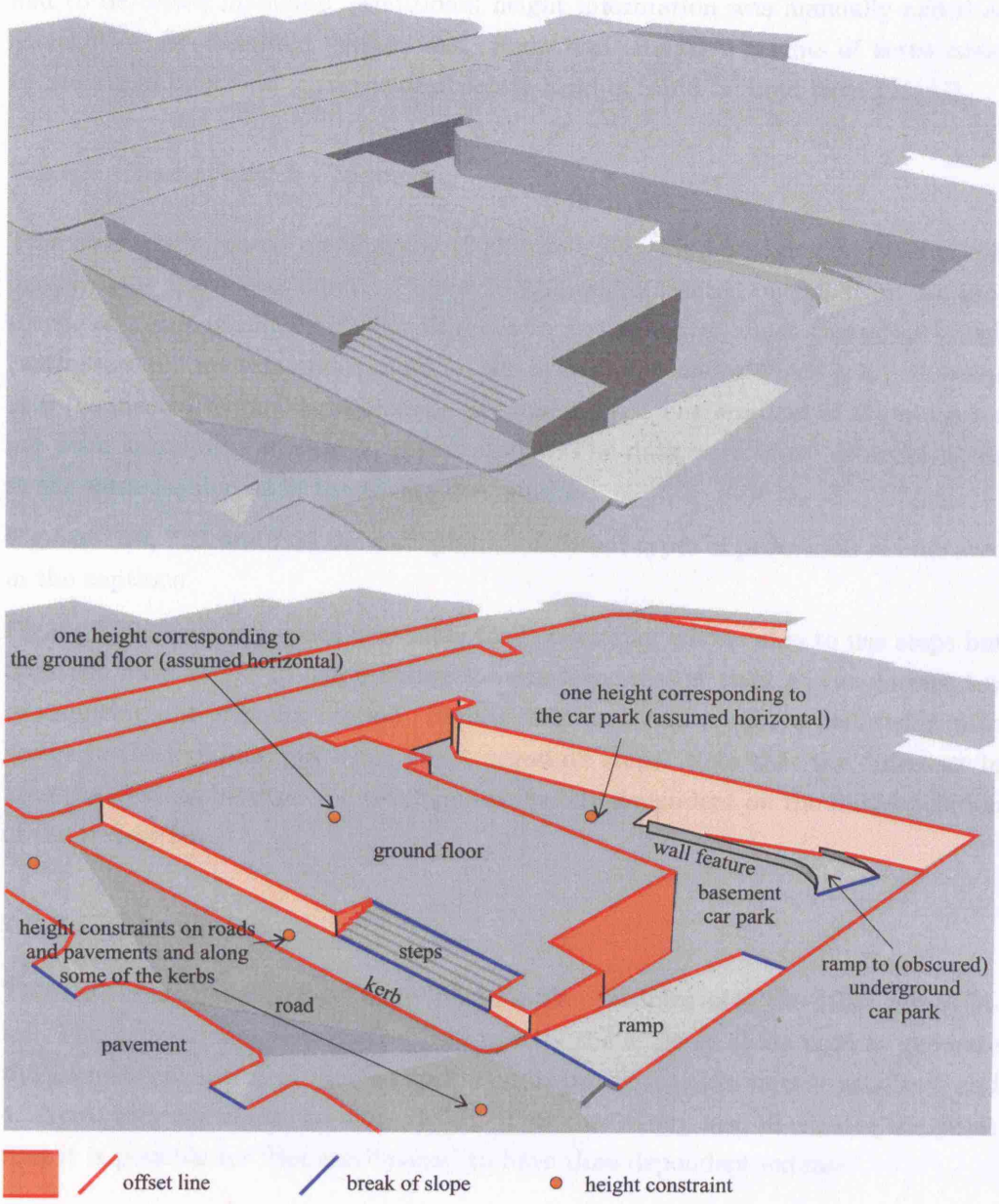


Figure 7.18: Illustration of the way in which the foundations of a building (1-19 Torrington Place, London, WC1E 7HB) is modelled using the framework. It shows the importance of offset lines and breaklines and illustrates how heights are incorporated. Heights are incorporated as absolute or relative height nodes within the layers. The kerbs are offset lines, some of which have heights specified along their lengths. The upper image is of a 3D Shapefile output by the prototype; the lower image has been annotated.

In terms of data collection, OS MasterMap identifies lines which are in between roads and pavements; most of these can be considered to be kerbs. These were automatically transformed as 'LineOffs' by the prototype, but they rarely formed completely closed areas suitable for the automatic generation of patches. They had to be closed manually. Additional height information was manually added as 'LineOffValues', heighted 'Nodes' and 'NodeOffs'. Relative heights of kerbs could be measured by a field surveyor or absolute heights could be used from LIDAR.

7.2.6 Case study 6 – access

This case study (based on Slingsby (2005) and Slingsby and Longley (2006)) will demonstrate the access model. Figure 7.19 shows annotated output from the prototype of a scenario of a building adjacent to a street, over which there is a bridge (with steps on one side and a ramp on the other), and under which is a pedestrian area (connected by one flight of steps and one ramp). The *gradient* of the ramp has not been taken into account in this example. The data were input as seven layers in the same fashion as in the earlier case studies.

Figures 7.20, 7.21 and 7.22 show outputs for different types of pedestrian as indicated in the captions.

Figure 7.20 shows the space accessible to a pedestrian who is able to use steps but does not have access to the building for whatever reason (lack of the correct key or requiring out of hours access). Figures 7.21 and 7.22 are for a pedestrian with access to the building, but who cannot negotiate steps. Note that the difference in accessible extent between the two figures is purely dependent on the initial position of the pedestrian.

7.2.6.1 Discussion

This case study demonstrates many of the same principles as in the other case studies. The pedestrian access routines are exactly the same as those used to generate the geometrical extent of the 'bspace1' and 'bspace2' features in case studies 1 and 4. Here, they are shown in more detail. This case study also illustrates the point that it is possible for 'BoundedSpaces' to have time-dependent extents.

7.2.7 Case study 7 – underground

Output from this case study is illustrated in figure 7.23. This case study shows that with the same concepts, a complicated array of underground spaces and tunnels can be described.

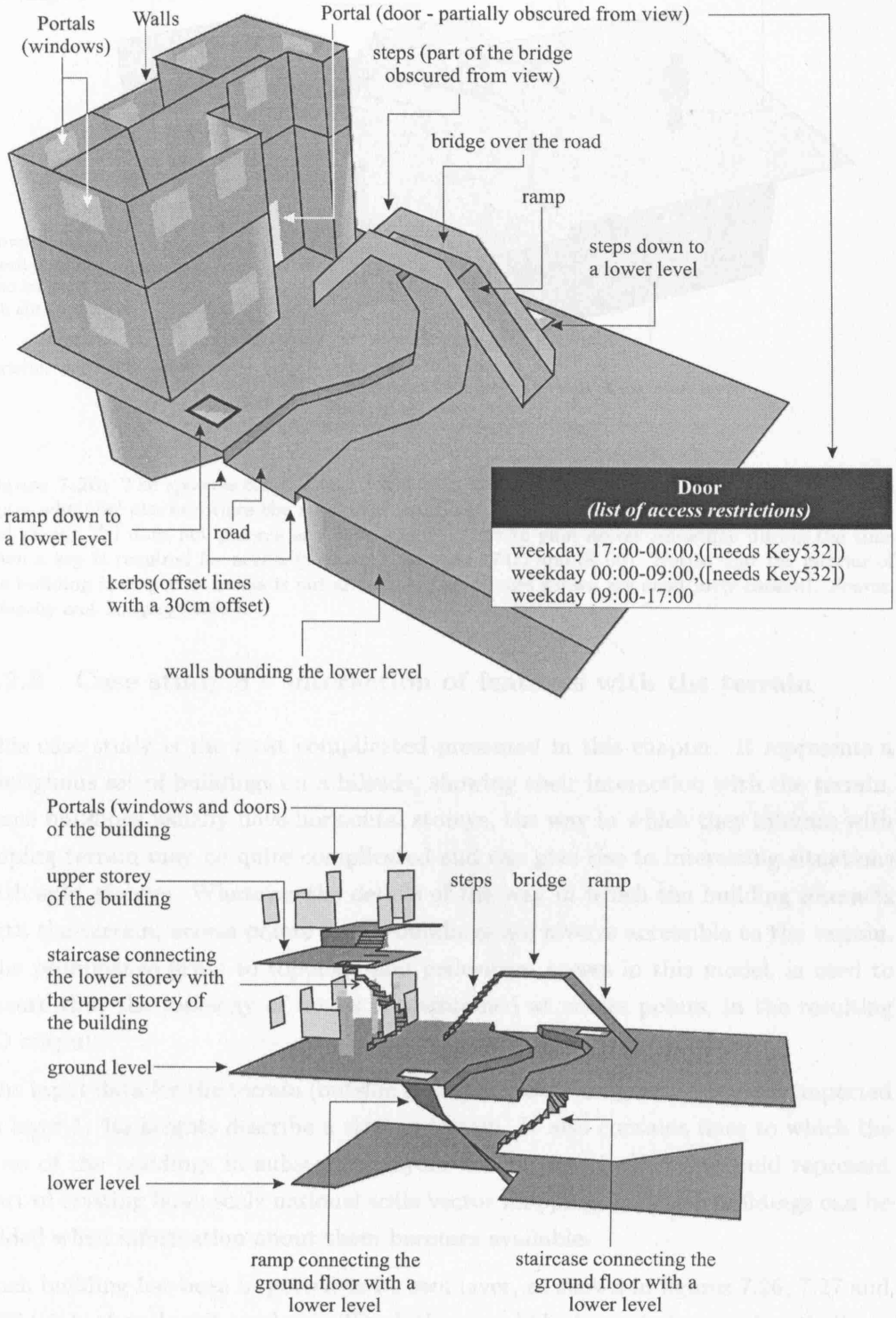


Figure 7.19: 3D output of a (fictitious) building next to a road with a bridge over it with an underground area. The lower image shows the same scenario with the walls removed. Only one of the doors ('Portal' features) has access restrictions attached; these are shown in the figure. (The colours shown are arbitrarily chosen). *Source: Slingsby and Longley (2006).*

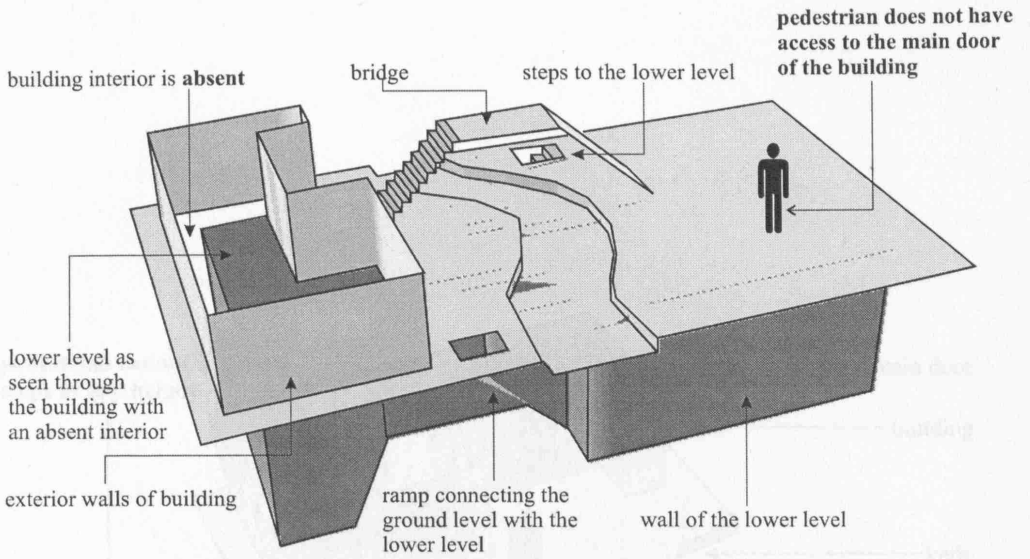


Figure 7.20: The space accessible to a pedestrian who starts from the position indicated in the figure who: *(a)* can negotiate the steps (its 'maxStep' value is greater than the height of the kerb and steps), *(b)* does not possess any keys and *(c)* tries to gain access sometime during the time when a key is required for access (weekdays between 17:00 and 09:00). Notice that the interior of the building is empty as access is not allowed. (The colours shown are arbitrarily chosen). *Source: Slingsby and Longley (2006).*

7.2.8 Case study 8 – interaction of features with the terrain

This case study is the most complicated presented in this chapter. It represents a contiguous set of buildings on a hillside, showing their interaction with the terrain. Since buildings usually have horizontal storeys, the way in which they interact with sloping terrain may be quite complicated and can give rise to interesting situations with split storeys. Whatever the details of the way in which the building interacts with the terrain, access points to the buildings are always accessible to the terrain. The prominence given to topology and pedestrian access in this model, is used to ensure that the topology of access is maintained at access points, in the resulting 3D output.

The input data for the terrain (outside space) is shown in figure 7.25 and is imported as layer 1. Its heights describe a sloping terrain. It also contains lines to which the lines of the buildings in subsequent layers will be attached. This could represent part of existing large-scale national scale vector mapping, to which buildings can be added when information about them becomes available.

Each building has been imported as its own layer, as shown in figures 7.26, 7.27 and 7.28 (since they do not overlap in 2D, all three could be imported as one layer). Since they are on a separate layer to the terrain, the edges of the building are topologically joined to geometries in input layer 1, all as offset lines. At access points, a height offset can be specified. However in this case, they have been imported as simply 'Lines'; since they are imported as coincident lines, the framework will treat them

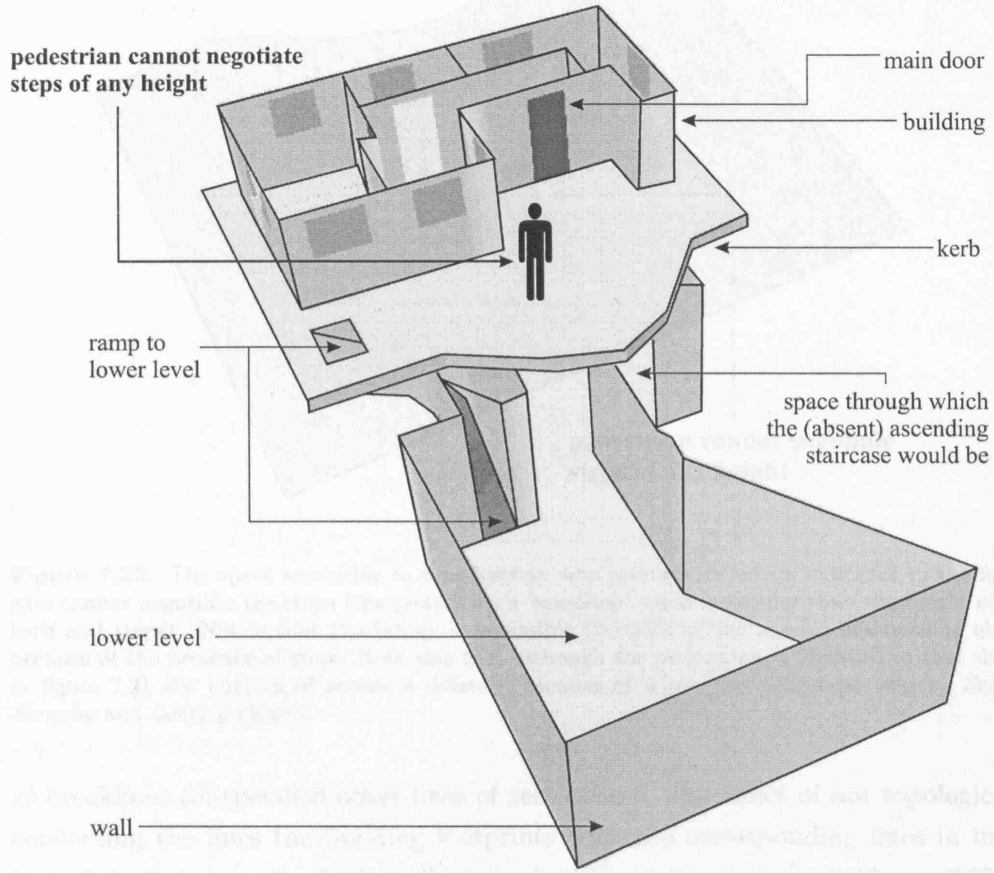


Figure 7.21: The space accessible to a pedestrian who starts from where indicated in the figure who cannot negotiate the steps (the pedestrian's 'maxStep' value is smaller than the height of the kerb and steps) and either *(a)* possess the correct key to gain access to the building ('key532') and tries to gain access during the time when the key is required (weekdays 17:00-09:00) or *(b)* does not possess a key and tries to gain access during weekdays between 09:00 and 17:00. Notice that access is allowed to the space below ground because of the ramp (ignoring the fact that it is perilously steep) and the lower storey of the building has access, but there is no access to the road and other side of the road (due to the kerb), the bridge (due to the steps on that side) and no access to the upper storey of the building (due to the flight of steps). *Source: Slingsby and Longley (2006).*

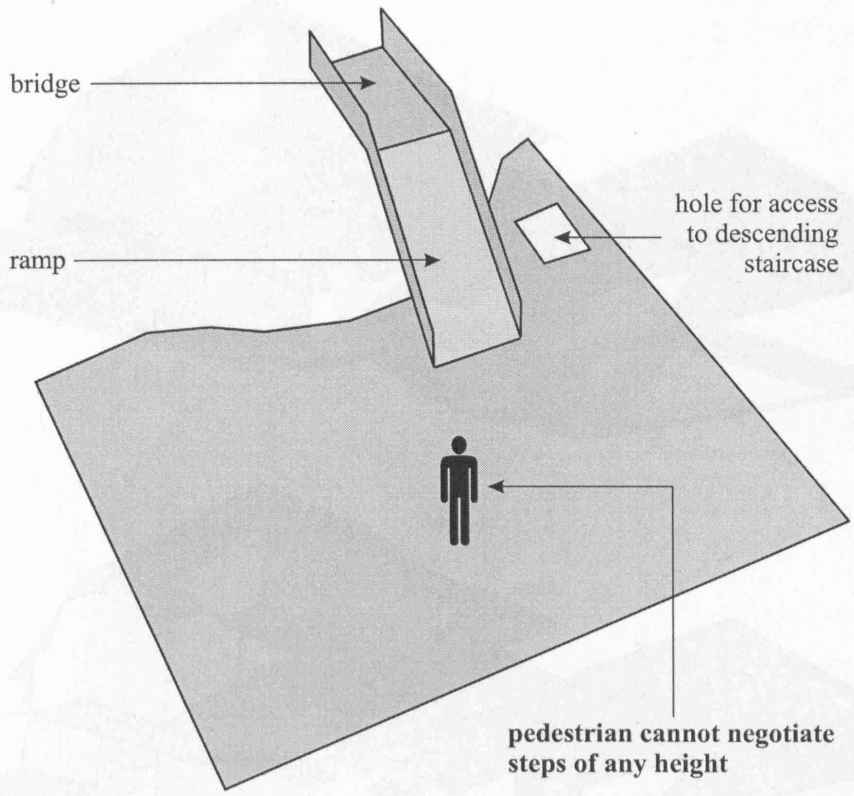


Figure 7.22: The space accessible to a pedestrian who starts from where indicated in the figure who cannot negotiate the steps (the pedestrian's 'maxStep' value is smaller than the height of the kerb and steps). Notice that the bridge is accessible (because of the ramp), but nothing else is because of the presence of steps, Note also that although the pedestrian is identical to that shown in figure 7.21 the pattern of access is different because of where the pedestrian starts. *Source: Slingsby and Longley (2006).*

as breaklines (of specified offset lines of zero offset). The effect of not topologically connecting the lines the building footprints with the corresponding lines in input layer 1 is shown on the back wall of the building input in layer 3 (figure 7.27) is shown in figure 7.31 – a space under the building has been created.

These three fictitious buildings are designed to illustrate three different situations of building on a slope.

The building on the left in figure 7.31 depicts a split level buildings with two access points at different heights, but heights which are consistent with the terrain in which the building sits. Each split level has one piece of height data taken from the access point, thus it assumes each split level is horizontal and the relative difference between the split levels is equal to the difference in access point heights.

The building in the centre is on one level, and it height is again taken from the access point. The offsets lines around the building give the vertical offset of varying amount around the building.

The building on the right of figure 7.31 shows that it is possible to have a building with a sloping floor. An addition piece of height information which enforces a

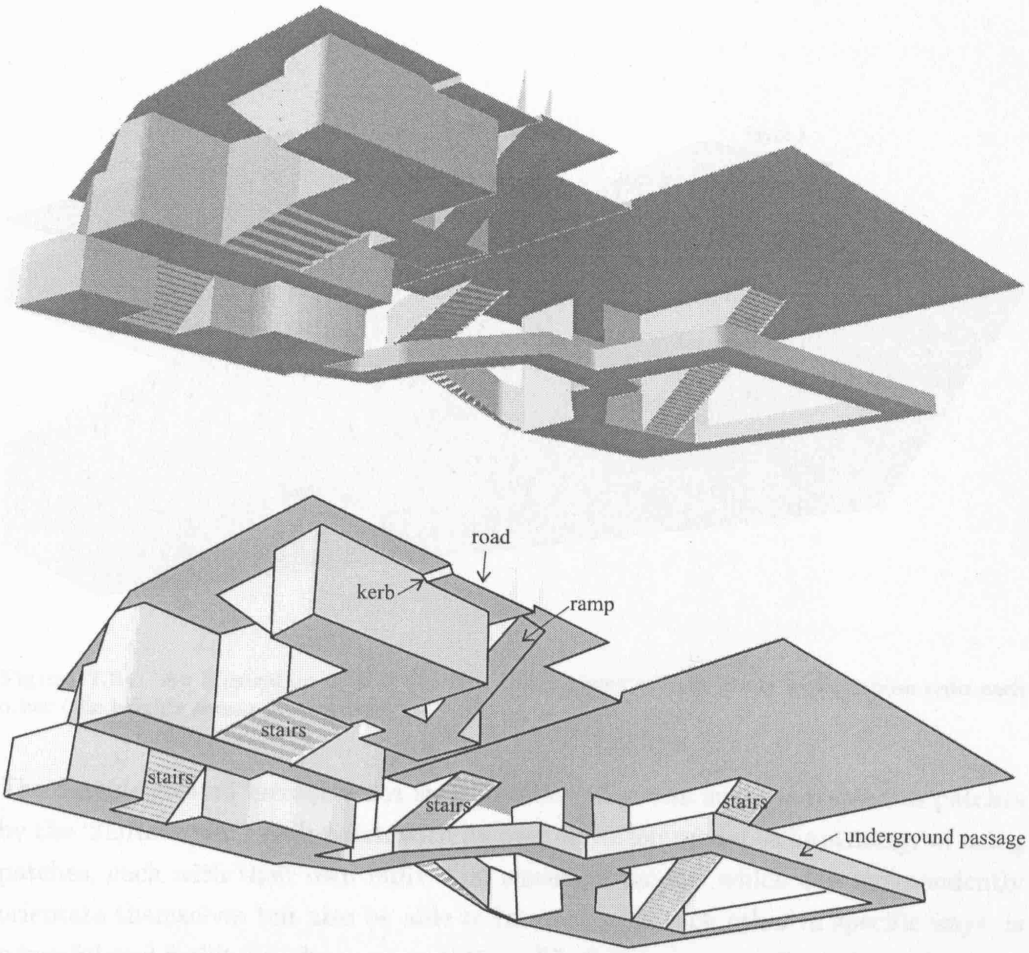


Figure 7.23: 3D output of case study 7, viewed from below. This can be input in two layers and shows a complicated set of underground spaces and tunnels, imported using input files similar to the other case studies.

separation between the terrain below and a position in the building results in a sloping floor. The topological disconnection between the lines at the back of the building and the terrain results in a space below the building.

7.2.8.1 Discussion

The main purpose of this case study is to show the interaction of buildings with the terrain model in which they sit. This is an important issue for any large-scale 3D model of buildings in the built environment (Stoter *et al.*, 2002, chapter 9). For full 3D models, all vertices/nodes can be given full 3D positions. Most 2.5D models have the concept of 'terrain' and 'buildings' and place the buildings on the terrain. This often presents problems when the terrain is very undulating. Stoter *et al.* (2002, section 9.3) and Rousseaux (2003) experiment with enriching digital elevation models of the terrain with the edges of features to produce better interpolations.

One of the novel aspects of the conceptual model presented here is the seamless treatment of interior and exterior space, a point also illustrated by this case study.

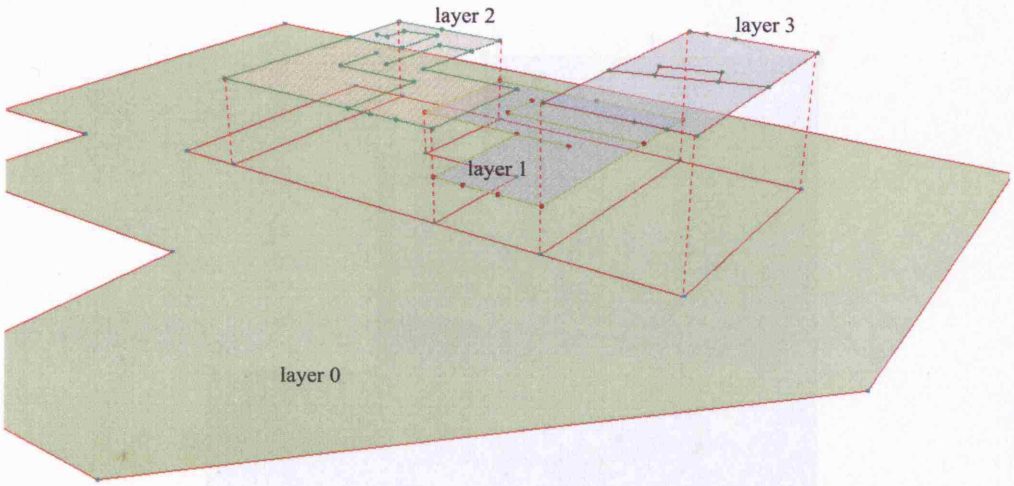


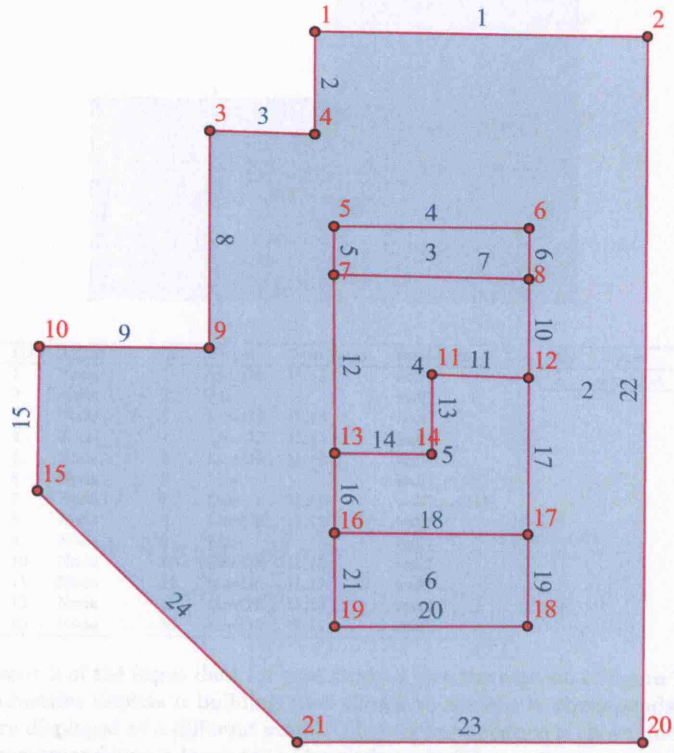
Figure 7.24: An illustration of how the four input layers of case study superimpose onto each other (the heights shown are arbitrary).

The outside ground terrain is not treated differently – all space is treated as patches by the '3DReasoner', each patch with its own elevation model. The strategy of using patches, each with their own individual elevation models which can independently orientate themselves but also be able to interact with each other in specific ways, is powerful and facilitates the representation of buildings on complicated terrains with minimal additional data.

Although not directly illustrated, data can be *input* in different numbers of layers, but in most cases, they will all result in the same data being *stored*; thus the same patches generated for output and equivalent outputs. The input data for all three buildings could alternatively be input as one layer. The offset lines between the buildings would still result in the '3DReasoner' splitting them into three patches (each with its own separate elevation model). Another option is that the two buildings on the left of figure 7.31 could be input inline with the rest of layer 1. The '3DReasoner' would still treat them as separate patches because of the surrounding offset lines, but additional breaklines would need to be provided in order to close the building polygon at the access points. Whichever of these input options are used, the stored geometries will form a seamless set of topologically-connected geometries.

7.2.9 Case study 9 – spiral ramps and staircases

This final case study shows how a spiral ramp or staircase can be input into the prototype. Because each input has its 2D topology built automatically, it must have no geometries which overlap in 2D. Since the spiral itself overlaps itself in 2D, it must imported in more than one layer (two layers are sufficient for this example).

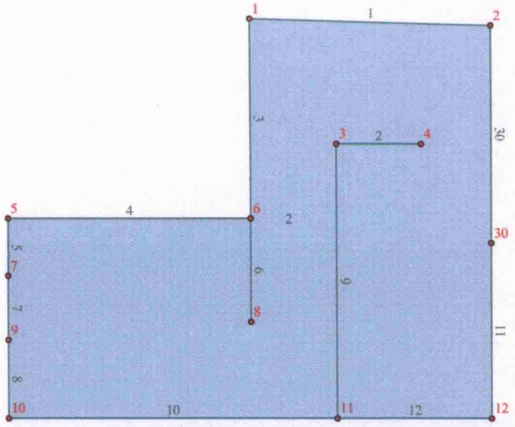


ID	Type	Ht/Off
1	Node	
2	Node	44
3	Node	44
4	Node	
5	Node	
6	Node	
7	Node	
8	Node	
9	Node	45.5
10	Node	43
11	Node	
12	Node	
13	Node	
14	Node	43
15	Node	
16	Node	
17	Node	
18	Node	
19	Node	
20	Node	40
21	Node	

ID	Type
1	Line
2	Line
3	Line
4	Line
5	Line
6	Line
7	Line
8	Line
9	Line
10	Line
11	Line
12	Line
13	Line
14	Line
15	Line
16	Line
17	Line
18	Line
19	Line
20	Line
21	Line
22	Line
23	Line
24	Line

ID	Type	Features
2	Polygon	bspace3
3	Polygon	
4	Polygon	
5	Polygon	
6	Polygon	

Figure 7.25: Layer 1 of the input data for case study 8 (see the caption of figure 7.1 for notation). This set of geometries represents a small area which is sloping towards the bottom right (see the values of the heights). The building outlines exist for the subsequent attachment of the buildings in the subsequent layers (figures 7.26, 7.27 and 7.28). Polygon 2 is part of the 'bspace3' feature defined in figure 7.29.



ID	Type
1	Node
2	Node
3	Node
4	Node
5	Node
6	Node
7	Node
8	Node
9	Node
10	Node
11	Node
12	Node
30	Node

ID	Type	Topology	Features
1	LineOff	l1_11	wall7
2	Line		wall7
3	LineOff	l1_13	wall7
4	LineOff	l1_14	wall7
5	LineOff	l1_16	wall7
6	Line		wall7
7	Line	l1_16	wall7,portal8
8	LineOff	l1_16	wall7
9	Line		wall7
10	LineOff	l1_18	wall7
11	LineOff	l1_17	wall7
12	LineOff	l1_18	wall7
30	LineOff	l1_17	wall7

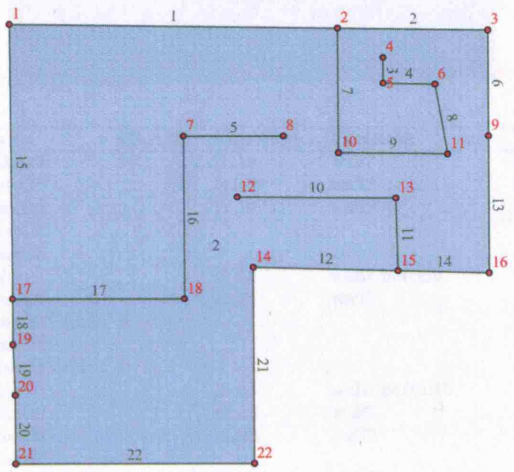
ID	Type
2	Polygon

Figure 7.26: Layer 2 of the input data for case study 8 (see the caption of figure 7.1 for notation). This layer of geometries depicts a building (not shown to scale). It corresponds to polygon 5 in layer 1 (this figure displayed at a different scale). This correspondence is shown in figure 7.24 which shows that this correspondence is by virtue of its position in 2D coordinate space. The surrounding lines are topologically linked to the corresponding geometrical primitives in layer 1. The reason for this is explained in the main text. All the lines form part of the geometrical extent of the 'wall7' feature shown in figure 7.29. The geometrical extent of 'portal8' is provided solely by line 7.

Layers 3 and 4 shown in figures 7.34 and 7.35 provide portions of the spiral. Line 10 in layer 3 has an attribute which links the two components of the spiral together. When the spiral ramp (or staircase) is built, it is treated as one unit whose combined 3D geometry is managed by a stack of patches as explained in section 6.11.3.3 and figure 6.59. The surfaces at the top (layer 1, figure 7.32) and bottom of the spiral (layer 2, figure 7.33) are heighted through their topological connection to the layers above and the below.

7.2.9.1 Discussion

Spiral ramps and stairs are commonly present in the built environment. This case study shows that these can be represented, using the same principles as apply to the rest of the model. However, the application of these principles does not only apply to spiral staircases or ramps, but any surface with no natural breaks which self-intersects in 2D. This case study shows that such surfaces must be arbitrarily split into chunks such that the geometries in each chunk do not self-intersect. Every topological join is treated as a breakline. For ramps and stairs, the way in which they are chunked makes no difference to the 3D output (as explained in section

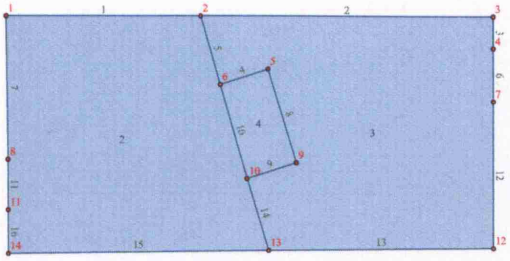


ID	Type	Ht/Off	RefGeom
1	Node		
2	Node		
3	NodeOff	5	p1_4
4	Node		
5	Node		
6	Node		
7	Node		
8	Node		
9	Node		
10	Node		
11	Node		
12	Node		
13	Node		
14	Node		
15	Node		
16	Node		
17	Node		
18	Node		
19	Node		
20	Node		
21	Node		
22	Node		

ID	Type	Topology	Features
1	LineOff	linkto0_7	wall6
2	LineOff	linkto0_7	wall6
3	Line		wall6
4	Line		wall6
5	Line		wall6
6	Line		wall6
7	Line		wall6
8	Line		wall6
9	Line		wall6
10	Line		wall6
11	Line		wall6
12	LineOff	l1_11	wall6
13	Line		wall6
14	LineOff	l1_11	wall6
15	LineOff	l1_12	wall6
16	Line		wall6
17	Line		wall6
18	LineOff	l1_12	wall6
19	Line	l1_12	
20	LineOff	l1_12	wall6
21	LineOff	l1_13	wall6
22	LineOff	l1_14	wall6

ID	Type
2	Polygon

Figure 7.27: Layer 3 of the input data for case study 8 (see the caption of figure 7.1 for notation). This is another building, corresponding to the position of polygon 4 of layer 1. However, when compared to that defined in layer 3, this building is not topologically linked to layer 1 along the right-hand side (lines 6 and 13). (This explains the gap seen in figure 7.31.) Additionally, node 5 is a relative height with respect to the 2D position of this node on polygon 4 in layer 1. It ensures that there is a separation between layer 1 and that point on the layer. Instead of a door ('Portal'), there is a break in the surrounding wall ('wall6') at line 19.



ID	Type
1	Node
2	Node
3	Node
4	Node
5	Node
6	Node
7	Node
8	Node
9	Node
10	Node
11	Node
12	Node
13	Node
14	Node

ID	Type	Ht/Off	Topology	Features
1	LineOff		l1_18	wall7
2	LineOff		l1_18	wall7
3	LineOff		l1_19	wall7
4	LineOff			
5	LineOff			
6	Line		l1_19	wall7,portal9
7	LineOff		l1_21	wall7
8	LineOffValue	0		
9	LineOff			
10	LineOffValue	0		
11	Line		l1_21	wall7,portal10
12	LineOff		l1_19	wall7
13	LineOff		l1_20	wall7
14	LineOff			
15	LineOff		l1_20	wall7
16	LineOff		l1_21	wall7

ID	Type	SubType
2	Polygon	
3	Polygon	
4	Polygon	Stairs

Figure 7.28: Layer 4 of the input data for case study 8 (see the caption of figure 7.1 for notation). This is the final layer of case study 8 and corresponds to polygon 6 or layer 1. It has an offset line through the middle with no height specified and a staircase between the polygons on either side. All the outer edges are topologically linked to the corresponding line primitives in layer 1. Apart from lines 8 and 10 which form breaklines at the top and bottom of the staircase, no other heights are given. Lines 6 and 11 (which also form the geometrical extents of 'portal9' and 'portal10') are topologically joined to the corresponding lines in layer 1, thus can take their heights from the polygons in layer 1. Thus each half of this layer (on either side of the breakline takes their heights from lines 6 and 11 which correspond to the heights of layer 1 at those positions. Since no other height information is given, each half is assumed to be horizontal. As can be seen in the 3D output in figure 7.31, the height of offset and the height of the stairs is dependent on the height of the terrain at 'portal9' and 'portal10'.

'Wall' Features			
Name	**Strength**	**ht**	**Cyclic existence**
wall7	secure	2.5	

'DoorKey' Features
Name
accesskey45

'Portal' Features					
Name	**Strength**	**ht**	**BaseHt**	**Cyclic existence**	**AccessRestrictions**
portal9	secure	2.5	0		everyday 0:00-23:59, needs accesskey45
portal10	secure	2.5	0		everyday 0:00-23:59, needs accesskey45

'BoundedSpace' Features		
Name	**Cyclic existence**	**Pedestrian**
bspace3		maxStep=0.5, has accesskey45

Figure 7.29: The features in case study 8

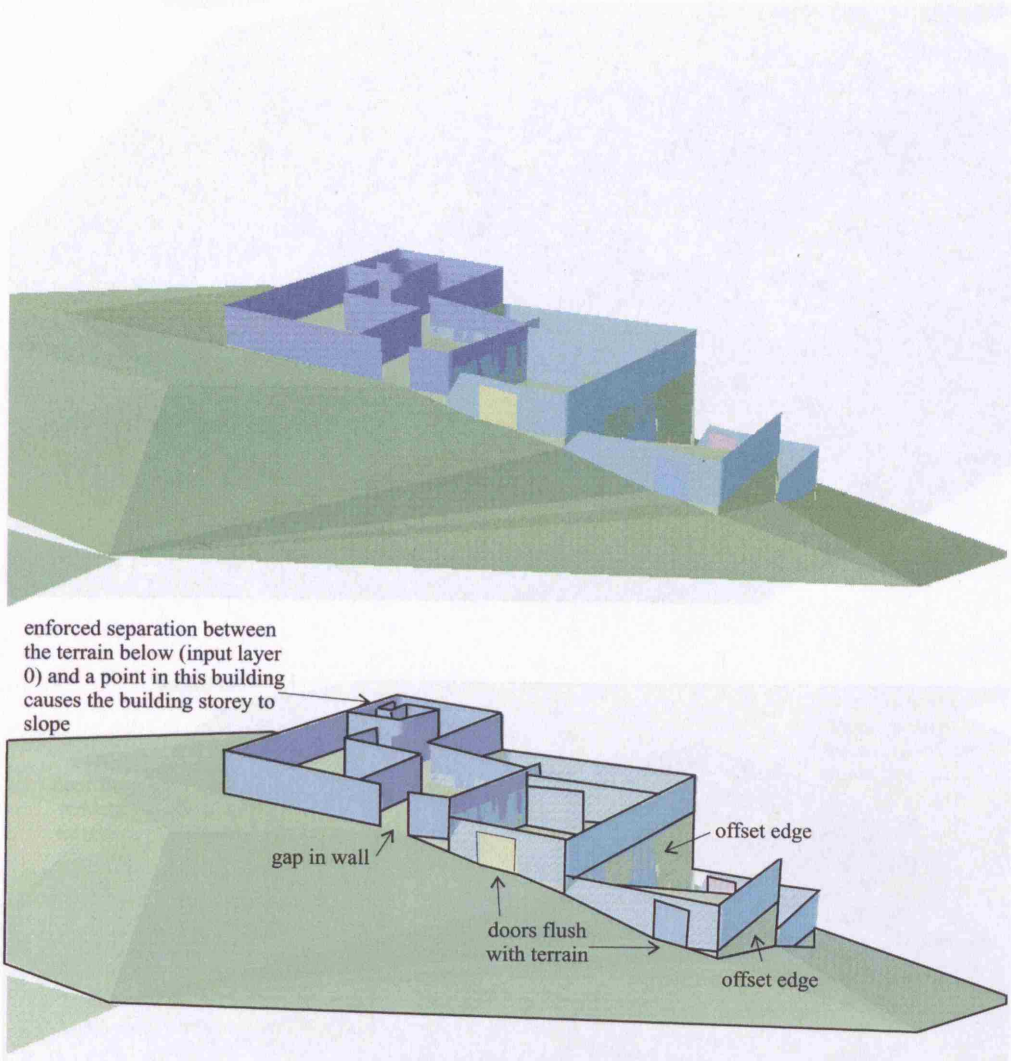


Figure 7.30: 3D output of case study 8. This shows how the contiguous buildings sit next to each other on the sloping terrain, taking their heights from the access points as defined in the input files. Notice that the building on the left is sloping upwards away from the page. Its elevation model is described by two heights on layer 3 (figure 7.27): the terrain height at line 19 and the relative height at node 3 (since node 3 is with reference to the polygon on layer 1, its height does not affect the height of layer 1's elevation model). (The input layers have been tagged onto some other input layers which explains the greater apparent extent of the input layer 0.)

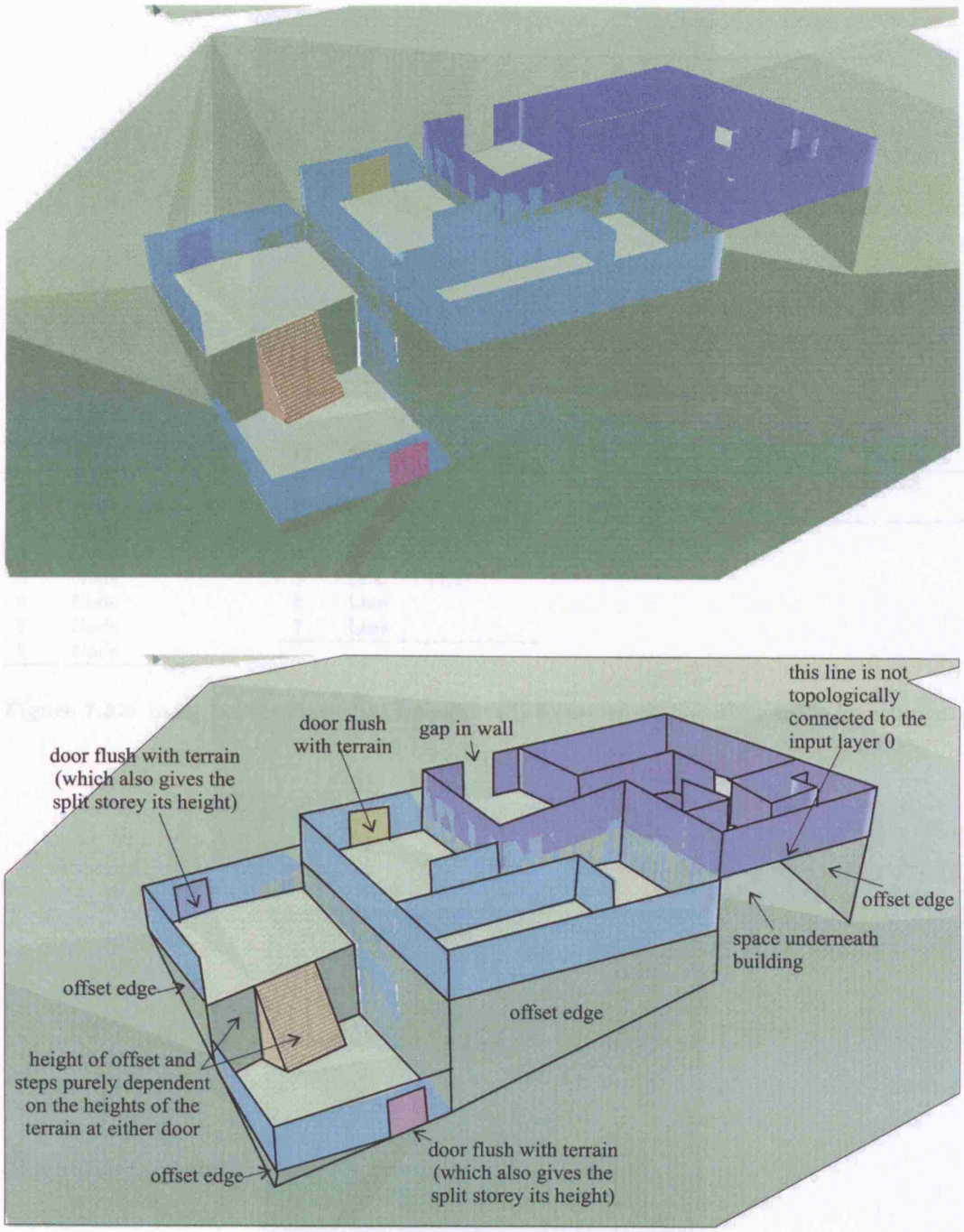
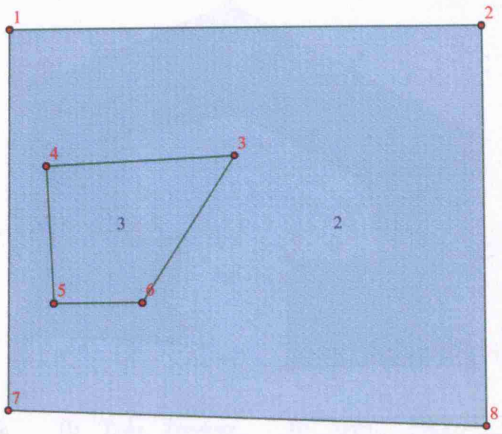


Figure 7.31: 3D output of case study 8 from a different angle. The building on the left (on the right in the other figures for this case study) shows the split level separated by an offset line where the offset is determined by the outside terrain height at the two doors and each level is assumed to be horizontal in the absence of other height information. The building is also horizontal, its sole height being the terrain height at the door. The building on the right (on the left in the other figures for this case study) shows that its slope deviates from the horizontal (as seen in the adjacent building) and shows the space underneath the building caused by the lines corresponding to the back wall not being topologically connected to the line primitives in layer 1. This figure also illustrates the varying offset height between the building storey and the terrain.

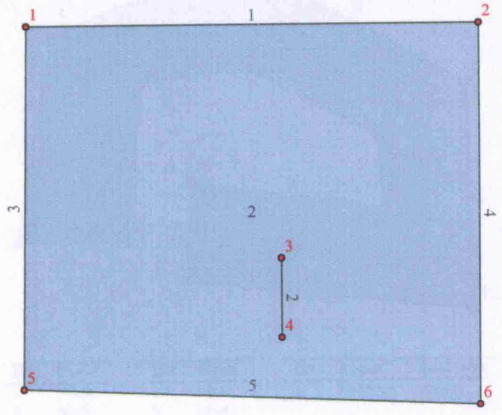


ID	Type	Ht/Off
1	Node	
2	Node	6
3	Node	
4	Node	
5	Node	
6	Node	
7	Node	
8	Node	

ID	Type	Topology
1	Line	
2	Line	
3	Line	
4	Line	
5	Line	l4_9
6	Line	
7	Line	

ID	Type	Subtype	Features
2	Polygon		space5
3	Polygon	Hole	

Figure 7.32: Layer 1 of the input data for case study 9 (see the caption of figure 7.1 for notation).

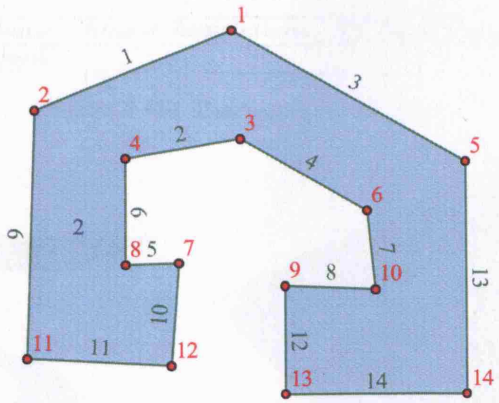


ID	Type	Ht/Off
1	Node	
2	Node	
3	Node	
4	Node	
5	Node	0
6	Node	

ID	Type
1	Line
2	Line
3	Line
4	Line
5	Line

ID	Type	Subtype	Features
2	Polygon	Ramp	space5

Figure 7.33: Layer 2 of the input data for case study 9 (see the caption of figure 7.1 for notation).

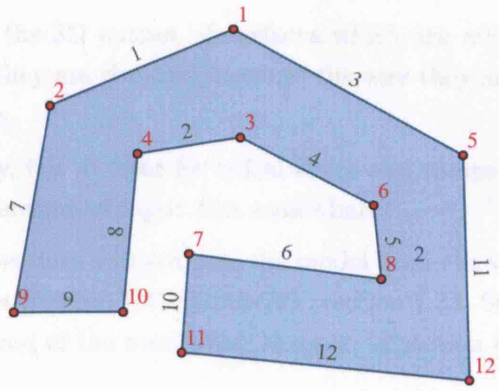


ID	Type
1	Node
2	Node
3	Node
4	Node
5	Node
6	Node
7	Node
8	Node
9	Node
10	Node
11	Node
12	Node
13	Node
14	Node

ID	Type	Topology
1	Line	
2	Line	
3	Line	
4	Line	
5	Line	
6	Line	
7	Line	
8	Line	
9	Line	
10	Line	l4_10
11	Line	
12	Line	l2_2
13	Line	
14	Line	

ID	Type	SubType	Features
2	Polygon	Ramp	space5

Figure 7.34: Layer 3 of the input data for case study 9 (see the caption of figure 7.1 for notation).



ID	Type
1	Node
2	Node
3	Node
4	Node
5	Node
6	Node
7	Node
8	Node
9	Node
10	Node
11	Node
12	Node

ID	Type
1	Line
2	Line
3	Line
4	Line
5	Line
6	Line
7	Line
8	Line
9	Line
10	Line
11	Line
12	Line

ID	Type	Features
2	Polygon	space5

Figure 7.35: Layer 4 of the input data for case study 9 (see the caption of figure 7.1 for notation).

'Space' Features		
Name	**Access Restrictions**	**Cyclic existence**
space5		

Figure 7.36: The feature in case study 9

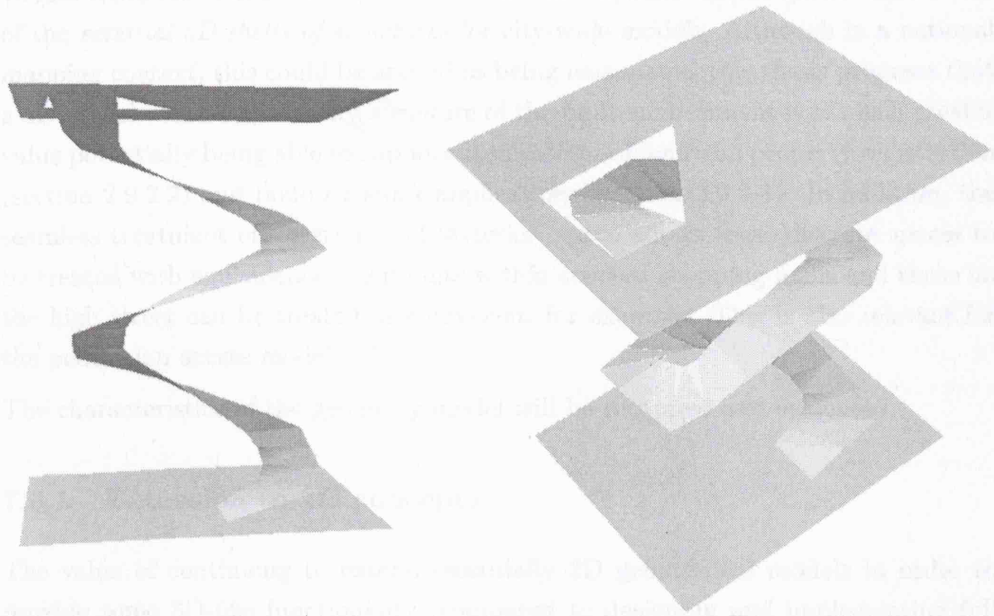


Figure 7.37: 3D output of case study 9, using the method described in section 6.11.3.3. (The appearance of slats is a function of the method used for interpolation for visual output).

6.11.3.3). However, the 3D output of surfaces which are not ramps or stairs, are affected by the way they are chunked, because the way they are chunked affects the position of breaklines.

This is rather clumsy, but at least for spiral stairs and ramps it should be possible to automate the generation of input files somewhat.

The following three sections will evaluate the model from the viewpoint of geometry (section 7.3), features (section 7.4) and access (section 7.5). Suggestions for further work are left to the end of the concluding chapter, in section 8.8.

7.3 Geometry

The conceptual model was intended to describe the structure of functional spaces within which human activity takes place. This includes spaces interior to buildings as well as exterior to buildings. The geometrical description to support this, is abstracted, such that the extent of the space is described but not necessarily its detailed geometry. Structures which do not describe spaces within which human activity takes place are out of scope; this includes roofs. In general, all such structures

are modelled through parameters on features, which themselves are represented by their 2D extent (footprint); such structures are described in section 7.4.

Section 3.2 presented the reasoning behind why the framework supports the seamless treatment of 'inside' and 'outside' space. As typified by many of the examples of virtual cities (section 2.9.1), there is usually an emphasis on the geometrical forms of the *external 3D shells of structures* for city-wide models. Although in a national mapping context, this could be argued as being reasonable, this thesis proposes that a 3D model which can identify *structure* of the built environment is of much greater value potentially being able to support applications of land and property registration (section 2.9.2.2) and building stock applications (section 2.9.2.3). In addition, the seamless treatment of 'interior' and 'exterior' space allows these discrete spaces to be treated with equivalence; shop units within covered shopping malls and those on the high street can be treated as equivalent, for example. This is also relevant for the pedestrian access model.

The characteristics of the geometry model will be recapped and evaluated.

7.3.1 Extension to 2D concepts

The value of continuing to extend essentially 2D geometrical models in order to provide some 3D-like functionality (compared to designing and implementing full 3D models in which all three dimensions are treated with the same priority and detail), is debatable. The extension of 2D models to achieve some 3D functionality is often seen as a solution which is barely adequate, temporary, compromising, but pragmatic. The comprise is between representational flexibility and the complexity of the data model and data requirement for its population.

In spite of the *potential* lack of representational ability, this thesis proposes a 2D-based model with extensions to encode 3D information which, it argues, makes the model adequate for fulfilling the requirements listed in chapter 4. These requirements are:

- compatibility with existing 2D data
- to be able to cope with incomplete height information
- the ability to abstract to 2D (for editing, the display of maps, display of floorplans and use in GIS and other 2D modelling software)
- to deal with pedestrian access (connectivity across surfaces)

The prototype based on the conceptual model is able to fulfil these requirements. It models geometry as a set of multiple layers representing ground or floor surfaces which implicitly represent spaces. Other structures are described through parameters on features, for example walls.

7.3.2 2D planar topology

2D planar topology information is embedded in the geometrical description. This explicitly defines the connections (in terms of pedestrian access) between spaces (though access may be impeded by the presence of certain 'Wall' and 'Portal' features). Since spaces are generally laterally accessible and a 2D-based model is used, this connectivity can be expressed with 2D planar topology.

In 2D topographic mapping, since the topography represents ground-based detail, it can be considered appropriate to drape it onto a terrain model; though as stated, this presents problems for overlapping structures such as bridges. This approach is applied to the model proposes in this thesis (in such a way that the latter point is addressed). Instead of topography being represented on *one* layer with a terrain model, it is represented across a topologically-connected set of multiple layers (patches), each of which has its own terrain model. These layers (different from the input layers) are automatically identified as *patches* (section 5.4.2) each with their own elevation model. Space is implicitly represented as the space above a patch. If there is a patch above, this will form the upper boundary of the implicit space below.

Within each layer (patch), *planar topology* is enforced (no overlapping geometries) because the geometrical primitives represent unique positions, lines and areas within the specific layer. These geometrical primitives are used to describe feature extents as illustrated in figure 5.19. 2D topology adjacency information is embedded in the geometrical model. These specific layers are then topologically connected where the surfaces meet and thus where access exists between them (again, this may be impeded or otherwise controlled by the presence of 'Wall' and 'Portal' features).

This topology between geometries which exists across multiple layers is an essential characteristic of the model, having implications for '3D reasoning' and for pedestrian access. For the '3DReasoner' the topology between layers acts as an essential height constraint to ensure that the reasoned 3D output is topologically consistent.

7.3.2.1 Discussion

Since the geometrical model essentially represents ground detail (in order to implicitly represent volumes), the use of a 2D geometrical model with the extensions proposed is appropriate. The presence of 2D topology and the requirement for the organisation of the geometry into distinct layers, enables the heights of the geometries within a layer to be related to each other. The fact that 2D topology within the distinct layers does not interfere with the topology in other layers allows spaces that overlap in 2D to be described. The topological joining of these layers enables the set of layers to be traversed, seamlessly being treated as one. Since layer surfaces represent the ground surface upon which pedestrians can walk, this surface topology can describe simple pedestrian access across and between layers.

7.3.3 Height constraints and surface morphology information

Surface morphology information about breaks of slope and vertical offsets, are embedded within the layers, and height information is scattered amongst the layers. These effectively 'pin down' points on the surface at specific positions. The heights of the geometries between these are interpolated such that they lie in the same layer as described by the embedded 2D topology.

Surface morphology information includes breaks of slope, steps and ramps. The addition of the offset line concept addresses one of the most obvious failings of 2.5D surface descriptions: the lack of representation of vertical morphological structures. These have been observed as of utmost importance particularly in the built-environment. It is also observed that along the length of a vertical offset, the amount of offset may be variable.

Heights act in a similar way to spot heights. Some are absolute heights (the 'AbsHeightNode' entity) and some of relative heights which describe their height with respect to another geometry (the 'RelHeightNode' entity). The latter can be used to describe the vertical separation as a point along an offset line.

7.3.3.1 Discussion: quantifying error in height

Figures 5.12, 5.13 and 7.44 show that there are three types of uncertainty in height data (assuming the heights in data points are certain, or within a specified uncertainty threshold):

- Certain – at height data points (within a specified uncertainty threshold)
- Uncertain – interpolated height (the degree is dependent on the distance from certain data)
- Very uncertain – extrapolated height (the degree is dependent on the distance from certain data)

Within these bands, the degree of uncertainty could be described as an inverse distance weighted function, centred upon measured height points.

Since relative heights can be specified (as well as absolute heights), there are situations in which although a relative height is known (i.e. the separation between two geometries), the absolute heights of these geometries are uncertain. This arises when the height of the reference geometry of the relative height is uncertain. The degree of uncertainty is dependent on the certainty of the absolute heights on which the reference geometry's is dependent. An example of where the absolute heights of the top and bottom of a relative height is *fairly* certain is shown in figure 7.44. In this figure, the closely positioned absolute height to the offset height on the left makes the absolute height certainty at the offset node almost certain.

If, in addition to this, if surveyed and specified height data could have their certainties quantified, the interaction of the different certainties would become very complex indeed.

Stairs, ramps, doors and lifts also affect height, as explained in section 8.8.2.3. The inclusion of these when quantifying uncertainty will add another level of complexity.

7.3.3.2 Discussion: conflicts

The 3D geometry is built from incrementally adding information to a TIN provided by height constraints and surface morphology information. It might be that height information from the different sources conflicts with each other. This might produce unrealistic and undesirable surface geometries. At present there is no filtering system for detecting such conflicts, but clearly such a system should be developed to help assess data quality (uncertainty). This is one of the identified roles of the 'MappingFramework' entity (illustrated in figure 5.2).

7.3.3.3 Discussion: multiple 3D solutions and sensitivity

The combination of having under-resolved height data available and the presence of relative heights leads to a situation where there may be multiple 3D geometry solutions, all of which honour the height constraints and surface morphology information.

Where height data are under-resolved it is assumed that the surface is horizontal. In order to assess whether there is any additional height information, all the patches whose heights are dependent on each other are searched as described in section 6.11.3.2 and the UML diagram in figure 6.55. If there is only one piece of height information, the patch is assumed to be horizontal and attempts to build the surface geometry of dependent patches are repeated. The order in which patches are assumed to be horizontal will affect the resulting 3D geometry as was illustrated in figure 6.61.

Figure 7.38 shows two outputs from the prototype, produced using the same data and the same rules, but where the order in which the patches were resolved was different. This results in a marked difference in the 3D geometrical output, yet the constraints have all been honoured (see section 6.11.3.5).

A related problem occurs if all the immediate heights are relative, rather than absolute. All the surrounding patches are searched, getting progressively further away, until at least one absolute height is found. The assumption of horizontality is applied and an attempt to build the patches' surface geometries is repeated. Another alternative absolute height a similar distance away which perhaps should be accommodated is not considered until that patch is searched. This situation – illustrated in figure 7.39 – should only arise where height data are unresolved and the assumption of horizontality needs to be made; however, this results in a worrying dependence on

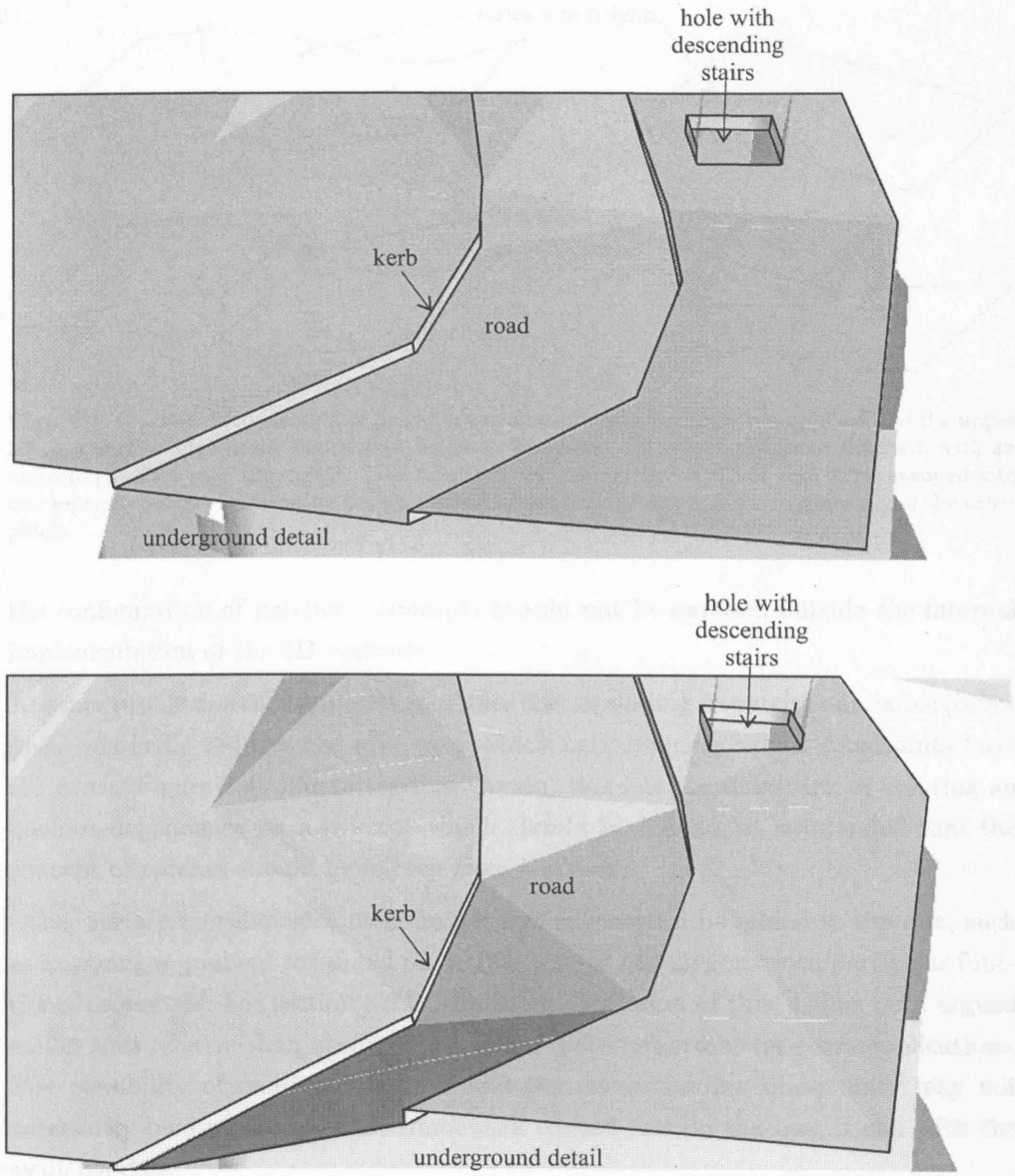


Figure 7.38: Multiple 3D solutions, depending on the order the patches were processed.

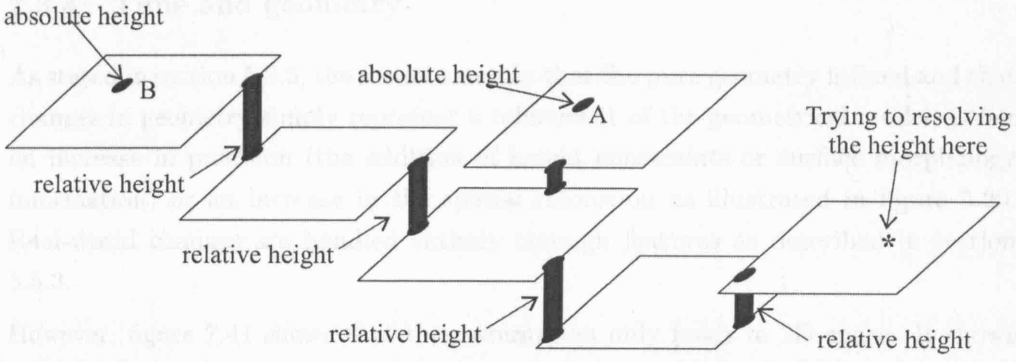


Figure 7.39: Illustration of an attempt to resolve heights (starting from the asterisk), where there are no local absolute heights. The algorithm needs to search adjacent patches. It will find absolute height A first, and height the asterisk with reference to this, assuming all the patches are horizontal (in the absence of other height information).

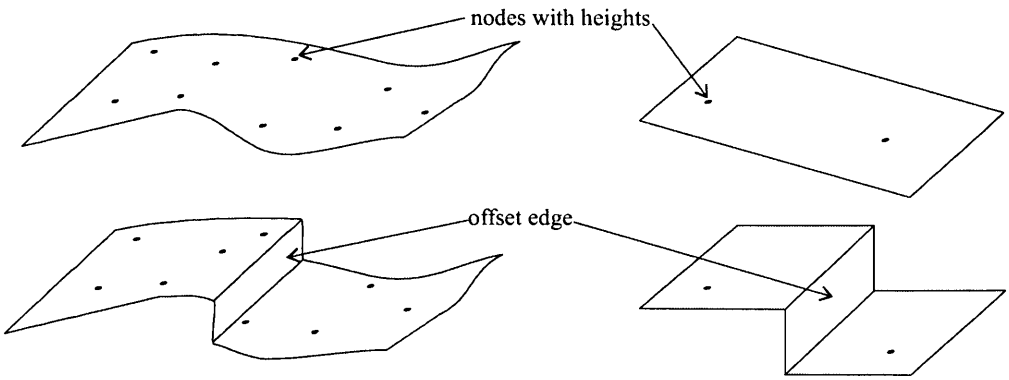


Figure 7.40: The disconnection of height constraint influence between two patches. On the upper left is a surface with many nodes with heights. The lower left shows the same situation with an unheighted offset edge through it. This results in the geometries on either side begin grouped into two separate patches, preventing the spot heights have any influence in the morphology of the other patch.

the configuration of patches – concepts should not be exposed outside the internal implementation of the 3D reasoner.

Another problem is that imposition of lines corresponding to patch boundaries (offset lines) markedly reduces the area over which neighbouring height constraints have influence. Figure 7.40 illustrates this. Again, this has the drawback of creating an obvious dependency on a concept which should be hidden (it is intended that the concept of patches should be hidden from the user).

Other surface morphological constraints and rules could be added to the mix, such as imposing a gradient threshold on certain groups of polygons with particular functional properties. See section 8.8.2.3 for more discussion of this. It has been argued earlier that relative than absolute height are more important for some applications. The possibility of multiple solutions and the corresponding uncertainty may not necessarily be a problems – the framework should just do the best it can with the available data.

7.3.4 Time and geometry

As stated in section 5.3.5, the model assumes that the pure geometry is fixed and that changes in geometry simply represent a refinement of the geometrical model; either an increase in precision (the addition of height constraints or surface morphology information) or an increase in the spatial resolution as illustrated in figure 5.20. Real-world changes are handled entirely through features as described in section 5.5.3.

However, figure 7.41 shows that this assumption only holds in 2D space. It shows that the framework cannot cope with representing morphological change, elevation change or the creation of holes though layers through time. This is because although space is exhaustively partitioned in 2D, it is not exhaustively partitioned in 3D. As

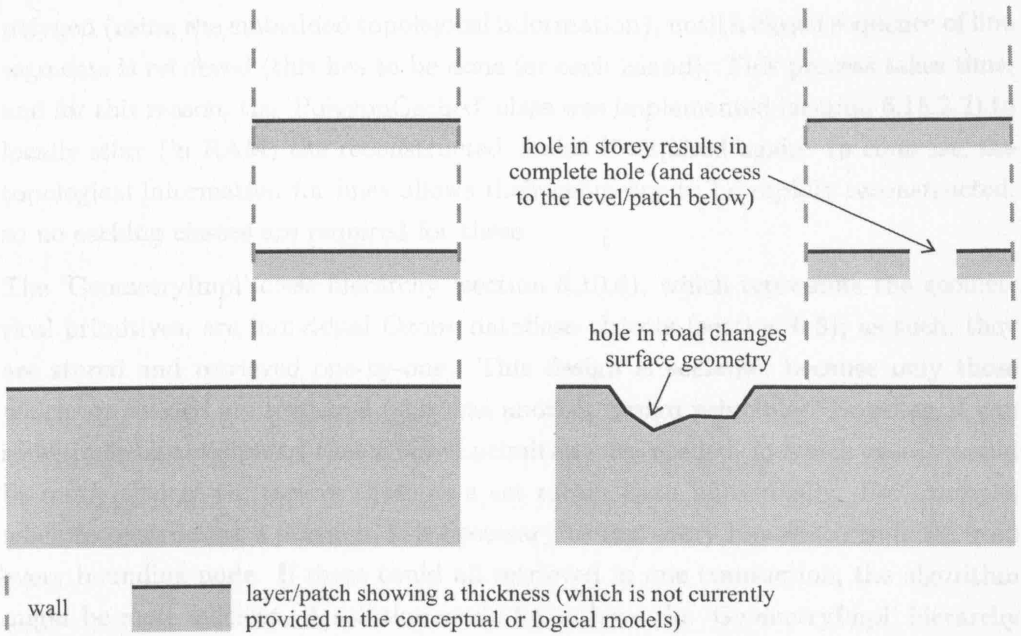


Figure 7.41: Cross-sectional illustration of possible changes in pure geometry. The hole in the road does not puncture the layer, so it result in a change in the surface geometry. The hole within the 'building' (although this figure only deals with pure geometry) does puncture the layer. The upper storeys of the building have also been reconfigured when comparing the situations on the left and right of this figure.

discussed in section 4.8, in general, an update process is *refine the state of the model* (as is the assumption for height constraints) or *to modify the state of the model* (not possible with this implementation). Since time is not taken into account for the geometry, no temporal queries can be performed and information about the past states of the database is lost. Possible solutions are discussed in section 8.8.2.

Another general issue with the way time is handled in the prototype, is it is assumed that change is instantaneous. This is inappropriate for modelling, for example, the natural processes of erosion, which, although slow, causes a change in the geometry of the environment.

7.3.5 Efficiency of the data structures

This implementation has been designed to be as close as possible to the conceptual model. This includes the requirement for minimal data storage; not to store information which is easily derivable. The implementation is purely proof-of-concept software. Neither the computational efficiency nor the actual implementation are of focus; rather the focus is on the conceptual ideas and the proof-of-concept. However, in this section, a brief appraisal of the efficiency of the implementation will be given.

Designing a model for storage efficiency is usually at the expense of the execution time. One of the slowest operations for constructing the 2D geometry is reconstructing a polygon, given its ID. A polygon is reconstructed by starting at a line segments bounding the polygon, and retrieving all connected segments which also bound the

polygon (using the embedded topological information), until a closed sequence of line segments is retrieved (this has to be done for each island). This process takes time, and for this reason, the 'PolygonCached' class was implemented (section 6.15.2.2) to locally store (in RAM) the reconstructed chains if required again. In contrast, the topological information for lines allows their geometry to be rapidly reconstructed, so no caching classes are required for these.

The 'GeometryImpl' class hierarchy (section 6.10.6), which represents the geometrical primitives, are individual Ozone database objects (section 6.3); as such, they are stored and retrieved one-by-one. This design is scalable, because only those which are needed are retrieved (this was another design principle). However, it can sometimes be anticipated that a set of primitives are needed, in which case it would be more efficient to retrieve these as a set rather than individually. For example, when reconstructing a polygon, it is necessary to test every line which radiates from every bounding node. If these could all retrieved in one transaction, the algorithm might be more efficient. A solution could be to have the 'GeometryImpl' hierarchy as standard Java objects and store them in 'chunks' in database object collections so that all the geometries in one 'chunk' can be retrieved at once.

The triangulation routines use the incremental point addition routine of Sloan (1987). Although this type of triangulation algorithm was found to perform poorly by Shewchuk (1996) when compared to other types, it was most suitable here because of the incremental nature of the building of the TINs. The classes used for the triangulation data model are in the 'GeometryPure' hierarchy, and use the same data model; triangles are modelled as polygons. A more efficient data model optimised for triangles should be used, which would make working with the TIN more efficient. Shewchuk (1996) found that implementing a data model designed for triangular rather than arbitrarily-sides polygons with islands was not only more space-efficient, but twice as fast (comparing the quad-edge model with his triangle model).

The 3D reasoner algorithms are by far the most time-consuming to run. As will be seen in the next section, this is dependent on the density and type of height information and its relationship with the other surface morphology data.

7.3.5.1 Discussion: performance and determinism of the 3D reasoning

An obvious consequence of the 3D geometry being 'reasoned' from a series of height constraints and surface morphology information is the length of time which it takes to build the 3D geometry. The time taken is dependent on the arrangement of height data. Poorly-resolved height data tends to lead to longer computation times because of the frequent fruitless searches of surrounding patches for morsels of height information. Another bottleneck is where relative heights are present. The majority of the case studies only had one absolute height value; the rest being relative heights. If a 3D geometry is built for an area a large distance away from an absolute height, the '3DReasoner' has to search adjacent patches in order to find one, before it can

build all the relative heights in between, to eventually reach an answer. As seen in section 7.3.3.3, the answers may be subject to a large amount of uncertainty.

7.4 Features

'Features' are conceptualisations of real-world objects.

7.4.1 Time and features

As stated, time is handled entirely through 'features' and the various facets of time are described in section 6.12.1.1. In order that the state of a database to be reconstructed at any point in turn, it is appropriate for time to be considered for all aspects of features; their existence, their attributes and their geometry (section 4.8). As acknowledged, this implementation only considers one facet of time, the time at which data are inserted into the database; section 4.8 lists others. In addition, only distinct snapshots in time and time periods defined by these are implemented.

As described in section 6.12.1.1, the existence of all features is recorded as one or more time-periods during which the feature is in existence. Some features also additionally define regular recurring periods when they are in existence. The addition and removal of every geometrical primitive for each feature is timestamped. This enables the geometrical extent of features to be reconstructed for any point in time. Some attributes of features are timestamped. This is true of the features with access restrictions – each addition and removal is timestamped so that the set of access restrictions for any point in time can be reconstructed. In this implementation, few of the attribute sets are timestamped. However, all could be. The modification of the 'AttribMap' class to incorporate time would add this facility to all the general attributes of features.

Cyclic or recurring time is used to define recurring periods in which features are in existence and also for the time-dependencies in the access permissions. As implemented, these are based on daily and weekly cycles, but clearly there are other lengths of temporal cycles.

The feature review section (section 4.6) reviewed research questions related to features. One question, related to features and time was the stability of features' identities through time. The framework proposed did not consider this; only how to represent features coming into existence and ceasing to exist (rather than the conditions under which this might happen).

7.4.2 3D geometry of features

As described in section 6.12, all of the 3D geometry is output through features. The geometry model (within the 'uk.ac.ucl.casa.aidan.phd.geometry' package) only deals

with topologically-connected layers of 2D geometrical primitives, which deal with the ground-based footprints of the ground-based geometry. Any geometrical extent above ground is described using parameters, allowing the geometries of significant real-world features to be described. Since the 3D structure of a feature is wholly parameterised and output by the specific feature class, this makes the model fairly extendable, as additional FeatureImpl classes can be defined.

An issue with features whose footprints are described by a point or line is a common geometrical abstraction for features in GIS. As can be seen from the case studies, many of the 'Wall' and 'Portal' features have footprints described as lines, which manifest themselves in 3D as infinitely-thin sheets. At the scales at which GIS usually operates, this is reasonable. When dealing with microscale spaces, this may become a problem.

Possible solutions within the current design are to have a 'thickness' attribute or to use polygons as the footprints.

7.4.3 Overlapping and nested features

As stated in section 5.3.1, the separation of the concepts of geometrical primitives from 'features' has two advantages. Firstly, it allows the pure geometry of environments (surfaces) to be modelled, without a dependence on real-world features. Secondly, it allows multiple definitions of overlapping features to geometrically overlap without the need to describe their geometries multiple times.

If every geometrical primitive 'knows' all the features for which it forms part of the extent, and if every feature 'knows' all the geometries which comprise its extent (as is the case in the logical model presented in chapter 6), then it is possible to easily analyse the relationships between features (such as whether there are feature hierarchies).

7.4.4 Access-based feature extents

Section 5.5.1.2 introduced the concept of features whose extent was access-dependent, for a particular pedestrian, at a particular time. The purpose of providing this type of feature is the observation that there appears to be a relationship between access and function in the built environment and that space can be grouped into subspaces by accessibility. No empirical analysis has yet been carried out, but examples of the important of pedestrian access are presented in section 5.5.1.2.

7.5 Access

The access model is designed such that an area of space can be delineated using information on geometry (slope and steps), attributes (access restrictions, door

strength, etc.), time (feature existence, geometry, and time ranges encoded in the access restrictions) and characteristics of a pedestrian. The features chosen and their characteristics are appropriate for describing pedestrian movement, but are by no means exhaustive.

Access at a specific time by a specific pedestrian is delineated in a piecemeal fashion by starting at a specific geometrical primitive, querying its geometry (including its 3D reasoned geometry), querying the feature of it forms part of the extent and then using the encoded topological information to pass to the adjacent geometries. This continues until the whole area for the specific pedestrian and time is delineated.

7.5.1 Representational ability

7.5.1.1 Gradient

As stated, the gradient of the terrain is not currently taken into account in the implementation for pedestrian access. If access is restricted to a fixed route, as is vehicular access on roads, the gradient along a line can be easily calculated. However, if there is some freedom to move across an inclined plane, pedestrians (or vehicles) have the freedom to not chose the path of steepest descent, but rather zig-zag across the surface. This will also increase the distance travelled. Such considerations would have to be taken into account.

7.5.1.2 The breaching of barriers

There are situations in which some access restrictions can be overridden or breached. Two examples are in emergencies and in burglaries. (In both cases, the pedestrian's 'MaxBreachLevel' will be elevated).

The access model supports this with the concept of a 'MaxBreachLevel' for pedestrians (section 5.7.1.1) and the corresponding concept of 'BarrierStrength' (section 5.7.1.4 and figure 5.2) for barriers ('Walls' and 'Door/Windows'). These concepts allow barriers to be categorised by their physical ability to deny access and allow pedestrians to be categorised according to the strength of barrier which they are able to breach. The specific classifications (figure 5.2) were designed to distinguish between a scale of unauthorised access:

- ignoring signage
- passing or moving a barrier without damaging it
- as passing or moving a barrier by damaging it
- overcoming a barrier using specialist equipment

In an emergency situation, if the official fire exits cannot be used, pedestrians are likely to leave the building by whichever means possible so their 'MaxBreachLevel' will be elevated.

In non-emergency and law/rule-abiding situations, the situation is more complicated. Examples of such situations are when a key or assess card has been left at home or if the access card reader is broken of simply does not recognise a valid card. In this situation, some pedestrians may be willing to breach a barrier, but this is likely to only be if it can be done without damaging the barrier. Some barriers are supervised, or have a means to contact someone for assistance. It may be useful for this fact to be recorded on barriers; i.e. the possibility for access permissions to be overridden.

7.5.2 Routing

As demonstrated, space can be delineated in a piecemeal fashion from geometrical primitive to geometrical primitive. For finding specific routes through space, many graph-based algorithms exist; thus the problem of generating routes can be reduced to generating graphs to be used by these algorithms. Usually, such algorithms use the topological information of the graph and the attribution attached to vertices and lines.

Dijkstra's algorithm is a well-known algorithm for computing shortest paths (or least cost paths) on graphs which are weighted by distance, cost or similar, a modified version of which is used by Lee (2004b). The approach of Lee (2004b) is to take a graph representing the connectivity of spaces ('combinatorial data model'), combine it with 3D geometry to form a 3D geometrical graph ('geometrical network model') and then run the modified version of Dijkstra's algorithm on it. (The 'combinatorial data model' in Lee and Kwan, 2005, represents other topological relationships as well).

In theory, all the data to generate a geometrical graph such as the 'geometrical network model' exist in the model. It should be possible to automatically produce a 3D geometrical graph with customised accessibility attribution using the methods implemented for use with a routing-finding algorithm.

7.5.2.1 Geometrical network graphs

Pure topological network graphs may be too much of a simplification for routing. Information about turning angle information (if it is assumed that access is along a linear route), height (e.g. overhead clearance under bridges in ITN), the width of gaps which may need to be passed through and the vertical gradient of the terrain, are all important. In a 3D context, Kirkby *et al.* (1996) takes into account the (vertical) gradient of the route in a 3D context and Lee (2004b) generates a full 3D geometrical network to incorporate connectivity between levels. In addition, when a route is visualised, it is usually plotted in Euclidean space as a line. For these reasons, it often necessary to generate geometry. This section considers this.

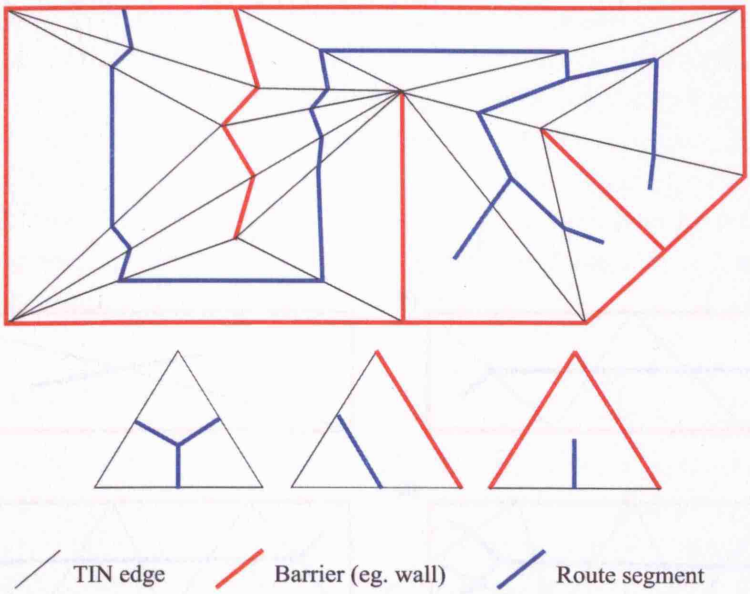


Figure 7.42: Illustration of a TIN-based algorithm for generating routes. The space is divided into triangles, with zero, one or two of their edges as barriers (red). Routes are drawn in blue according to the rules shown at the bottom of the figure. Each triangle effectively forms a very small component of the network where routes bisect all the edges through which access is allowed.

The 3D geometrical networks generated by Lee (2004b) are the centrelines of corridors connected to the centroids of spaces, in different storeys. The routes are generated using a modified medial axis transform ('straight-MAT'). These are used to build fully-connected networks upon which a modification of Dijkstra's algorithm is run. This technique does not work will polygons with islands.

Some small experiments were performed to assess the feasibility of using constrained Delaunay TINs for routing (these were implemented as a 'getRoute' method in the 'Space' entity), where barriers to movement form breaklines. These subdivide space into triangles with zero, one or two of the edges as barriers, in which inferences about the route can be made as shown in figure 7.42. A fully-connected network can be produced but it may have a rather strange geometry and is dependent on the triangulation. Figure 7.43 shows a range of alternative routes for different triangulations. The last two examples in the figure shows a possible effect of adding more vertices. The part of the corridor which is curved comprises many small lines which give rise to the Delaunay Triangulation shown and an unwanted branch in the network.

This approach was originally attempted for finding the centrelines of ramps and staircase in order to interpolate a surface which is horizontal along the route. The unreliability of finding an unbranched route and the occasional presence of unwanted kinks in the route was the reason why the approach was abandoned in favour of the simple approach which was eventually implemented (section 6.11.3.3). This simplistic approach produces an adequate surface, but relies on there being an entry and exit specified (as two breaklines).

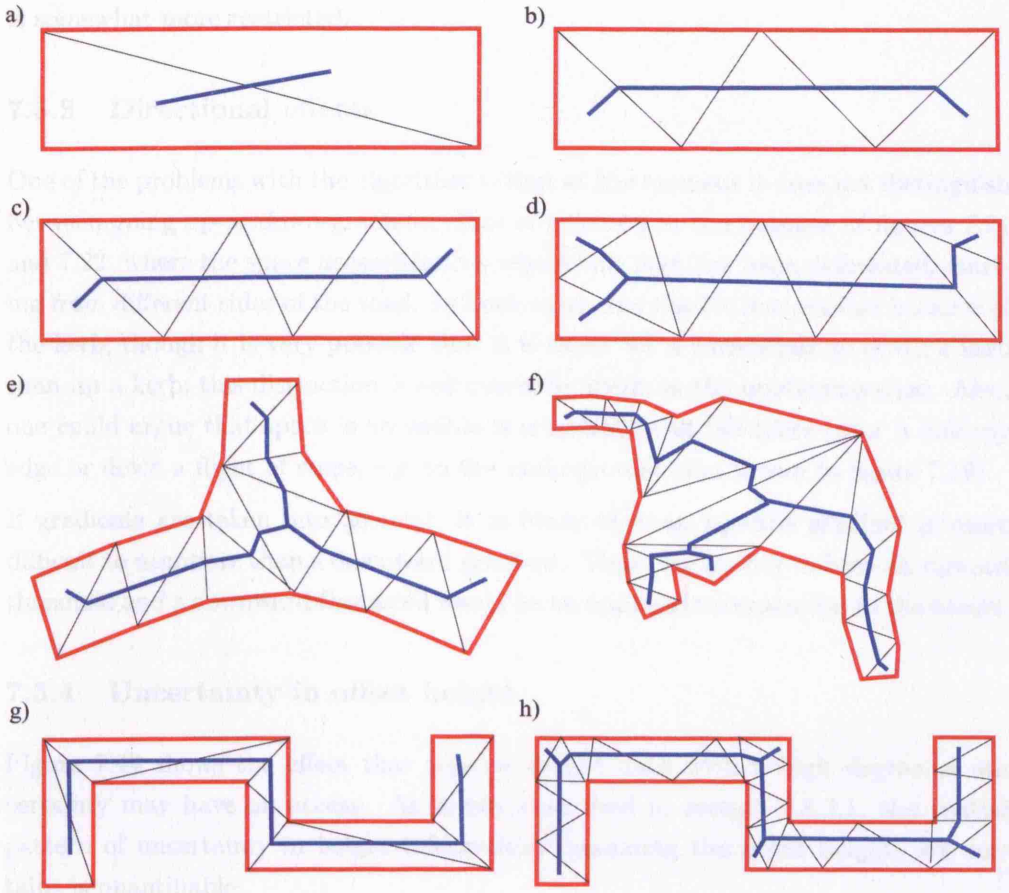


Figure 7.43: Alternative route geometries are strongly dependent on the triangulation. *(a)* The triangulation is only built from the corners of the rectangular polygons. *(b)* The triangulation is densified to that the long triangle edges are subdivided such that they have a similar length of the shortest edges. The result is that the route is better aligned along the long axis. *(c)* This illustrates that the route geometry is strongly dependent the TIN. *(d)* This illustrates that the route geometry is strongly dependent the TIN. *(e)* The route at a junction. *(f)* A corridor with an irregularly-shaped open space produces a fully connected network with a strange geometry. *(g)* A corridor with a triangulation only built from the essential vertices. *(h)* A less satisfactory route generated from a triangulation with many other vertices, showing the sensitivity of the route to the TIN.

Regnauld and Mackaness (2006) use this approach to generate river centrelines in order to generate the topology of whole river networks from contiguous polygons representing sections of river and lakes, cut by overpasses in 2D vector topographic mapping (a similar problem to that of road routing in 2D topographic maps). They distinguish between rivers and lakes by using geometrical rules. A similar strategy may be used to distinguish between open spaces (where routes are generally straight lines connecting the relevant entry and exit points) and corridors, where the route is somewhat more restricted.

7.5.3 Directional effects

One of the problems with the algorithm is that at the moment it does not distinguish between going up-or down-gradient. This is reflected in the outputs of figures 7.21 and 7.22, where the space accessible to a wheelchair user has been delineated, starting from different sides of the road. In both cases, the road is inaccessible because of the kerb; though it is very possible that it is easier for a wheelchair go down a kerb than up a kerb; this distinction is not currently made in the implementation. Also, one could argue that space is accessible is somebody can fall there (over a balcony edge or down a flight of steps, e.g. to the underground area shown in figure 7.19).

If gradients are taken into account, it is likely that an upward gradient is more difficult to negotiate than a downward gradient. Thus, the facility to have an upward threshold and a downward threshold would be an appropriate extension to the model.

7.5.4 Uncertainty in offset height

Figure 7.44 shows the effect that relative height data with a high degree of uncertainty may have on access. As briefly described in section 7.3.3.1, the spatial pattern of uncertainty in height information (assuming the input heights are certain) is quantifiable.

7.6 Summary

This chapter has evaluated the conceptual and logical models, firstly from the point of view of worked examples; and secondly, from the point of view of the three strands of this thesis. The worked examples illustrate different properties of the model, with an emphasis on representational ability. The first worked examples presented are accompanied by a full description of the input data, to demonstrate which data are required for specification. All the case studies have a full and detailed discussion of the properties of the model, including a description, an evaluation and an overview of potential problems. The subsequent evaluation and discussion of the model in terms of the three strands of geometry, features and access, is more technical in character.

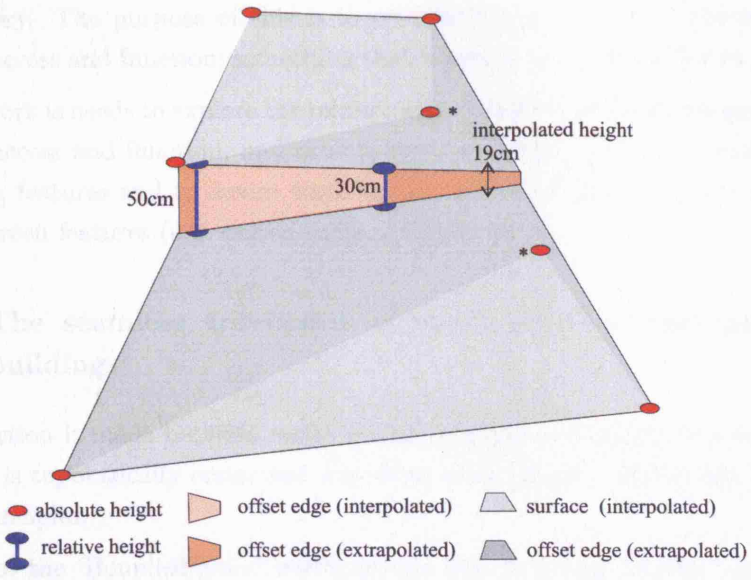


Figure 7.44: Uncertainty in vertical offset height. This example shows two relative heights (offset nodes). The extrapolated height is 19cm. This height is uncertain, because it is not directly based on a data point, rather it is extrapolated from closest offset node and the two asterisked absolute height. This potentially has significant implications for access for pedestrians capable of negotiating up to 20cm step heights.

It is the conceptual model which is the focus of this thesis. It is acknowledged that the implementation is slow and difficult to use and that a robust implementation may need to be implemented differently, but the realisation in the prototype (supported by the logical model) is a proof-of-concept which is relied upon in this in the evaluation (this chapter).

To summarise, this chapter shows that the model fulfils the design goals which were set in chapter 3.

7.6.1 The ability to describe different conceptualisations of features

As observed in section 7.4, the model presents a very stripped-down approach to dealing with features in so far as there is no built-in intelligence in feature definition. Features are prescribed by the data provider. Due to the real-world independence of geometrical primitives, multiple features can be defined which overlap and nest in different ways.

From a basic feature type (the 'Feature' entity) with a prescribed geometrical extent (described with a set of geometrical primitives) and an arbitrary set of attributes, a series of more complicated feature types can be defined. These have their own parameterised 3D geometry and are able to affect pedestrian access. In the implementation, the routines and behaviour which allow them to do this are self-contained; i.e. defined within the class without external supporting classes. In addition to the simple prescribed geometry set, one type of feature included is represented by the 'BoundedSpace' entity which takes its geometrical extent from a pedestrian access

based query. The purpose of this is to provide a tool to explore the relationship between access and function; something that was not explored further in this thesis.

Further work is needs to explore the relationships between different feature types and between access and function, and thus to explore the possibilities of automatically classifying features and to devise ways for the automatic identification of relationships between features (e.g. nested units of buildings).

7.6.2 The seamless treatment of space exterior and interior to buildings

No distinction it made between space within building and space outside buildings. All space is topologically connected including across layers. This fulfills one of the design principles.

The use of the 'BoundedSpace' entity is one way in which interior space can be defined; i.e. space which is bounded by a set of walls and specific doors. It it hoped that various precise definitions of 'interior and 'exterior' space can be used to automatically classify space as being 'interior and 'exterior'.

7.6.3 Pedestrian accessibility

Case study 6 (section 7.2.6) provides a simple example of the time- and pedestrian-dependent access model. Access is discussed further in section 7.5, including how linear routes can be extracted and their uses in graph-based routing algorithms. The conceptual model considers various aspects of access and it proposes a framework for representing these. It considers access in terms of geometry, time, characteristics of the specific pedestrian and the effect that a feature has on access (access permissions). This multivariate access model provides a rich description of the state of pedestrian access, embedded within a model of the built environment, at a spatial and temporal resolution which, it is believed, has not been attempted before.

The specific attributes suggested for the pedestrians and the access restrictions exist to illustrate the operation of the model. The addition of other attributes and dependencies may be appropriate for access in different contexts.

7.6.4 The concept of a data repository to which information can be added to in an incremental fashion

The framework is designed to take well-resolved 2D data, largely obtained from existing topographic vector mapping and a less well-resolved set of height and surface morphological information. Assuming the height data conform to the required minimum set (at least one height for every patch, at least one of which is an absolute height), a 3D geometry can be generated. As more heights and surface morphology information is added, so the generated 3D model will improve. Properties and

problems of the use of incomplete data have been identified, but these are associated with under or poorly resolved height data – a problem with any system relying on incomplete or ambiguous data.

It is also possible to improve the 2D spatial resolution by densifying the existing geometrical primitives.

7.6.5 Time

The history of change of features in the database is recorded in such a way that it is possible to reconstruct the state of features for any point in time as it existed in the database.

As discussed in section 7.4.1, the current model is quite simple in this respect. Firstly, no distinction is made between the real-world change in time and the time at which it was entered in to the database. Secondly, no distinction is made between the possible reasons for the change (error, refinement or real-world change). Thirdly changes over time are not considered – they are assumed to be instantaneous (section 7.4.1). Finally, changes in geometry are not considered – it is assumed that all changes in geometry represent refinement only (see section 8.8.2 for suggestions of further work).

Chapter 8

Conclusion and Further Work

This thesis has explored the issues of digital mapping in three-dimensional space in the context of national mapping. It has done the following:

- Identified a gap in data provision for GIS-type analyses of the built-environment.
- Explored the use and meaning of '3D' in a GIS context, a context in which structured 3D data on 'real-world' features is difficult to obtain and analysis software does not currently handle 3D data very well.
- Explored the issues around conceptualising 'real-world' features, including identity, uncertainty, semantics, time and inferences which can be made from properties of features.
- Explored the issues around the complexities of representing context-dependent pedestrian access.
- Proposed that there should be an integrated approach to storing geometry, features and access at the highest level of detail available, such that it can be transformed and generalised to support a wide range of applications.
- Proposed that uncertainty and the unavailability of data ought to be managed, providing the best output possible with the available data (supported by a quantifying report) rather than the requirement that fully-resolved data be made available or derived data stored.
- Proposed a novel design for the multilayered 2.5D storage of these data.
- Provided examples and outputs to test the properties, sensitivities and suitability of the proposed design.

These chapter will summarise the main points, and then will identify implications and areas for future work in section 8.8.

8.1 Gaps in data provision

Chapter 2 set out the context of the thesis. It explained that rich, well-defined digital models, underpinned by strong frameworks, are able to support a wide range of applications. The change from rather *ad hoc* and unstructured drawings to 'building information models' in architecture was presented as a case study. The changing character and role of national mapping was then reviewed, which identified the change from simple 2D map-based representations towards feature-based databases which encapsulate many aspects of the environment. Some differences in the nature of data supporting CAD-type (architectural and engineering) and GIS-type applications and analyses are discussed.

Following this, a set of applications were reviewed which are not supported by the combination of 2D national mapping data and 3D building information models. In terms of scale, national mapping data are 2D and cover large areas of external space, whereas building models are detailed and precise 3D models, modelled in isolation from their surroundings (in particular, applications requiring 'inventories of building and property units', as described in section 2.9.2, cannot be supported by the current dichotomy between national mapping and individual detailed building models). As well as the differences in scale, there are differences in the way features are conceptualised. Current data products present their own feature types which are unlikely to be universally fit for use and are unlikely to be directly comparable with others. Emergency planning in a city context was also identified as an application domain with similar data needs, with the addition of pedestrian access. Other applications involving pedestrian access in, around and between buildings were discussed, as were traditional 'virtual city' models.

8.2 The meaning of '3D' in a GIS context

It was noted (e.g. in section 2.9.1) that many imagine a 3D mapping product to be one in which the geometry of the outer shells of structures (particularly buildings) are placed on a terrain model. The review of the analysis of buildings (section 2.9.2) showed that 'buildings' consist of sets of functional spaces which exist in three dimensional space, the configuration of which describes the built environment. This thesis proposes that 3D digital mapping should concentrate on the *structure of spaces* in the environment. This does not have to apply to buildings exclusively and can also apply to underground spaces, bridges, tunnels and caves, including spaces on the terrain surface traditionally modelled by existing maps.

Section 4.5.5 analysed the meaning of the term '3D', showing that data can be described such that its 3D geometry can be constructed and oriented in 3D without using a full-3D description. A full-3D description allows geometry to be equally described in terms of all three dimensions. 3D topology can be described indepen-

dently of 3D geometry and is required for full 3D topological queries. From the characteristics of GIS-type analysis, the difficulty in obtaining 3D information, the lack of support for 3D in GIS tools (section 2.6.2) and observations of the characteristics space, a 2.5D approach was chosen as appropriate, extended to address its traditional failings.

Although the 3D structure of the environment is of interest to GIS-type applications, GIS tools tend to be 2D-based. Thus, it should be possible to abstract the information into 2D for use in such tools.

8.3 Conceptualising 'real-world' features

Issues of boundary delineation, the identification of features, the properties of features, the classification of feature – the *essence* of different features – were discussed in sections 2.4 and 4.6. Time adds another dimension. These complicated and poorly determined issues are simplified to be dealt with here; for example, it was assumed that boundaries are sharp and are known in 2D (see section 8.8.2) and only one facet of time was used.

From the issues discussed, the need for a feature-based model capable of being able to store any feature was identified. It was assumed that the features have already been identified (their geometry and their properties). The importance of storing the properties of features was illustrated by the discussion of how buildings can be defined (section 4.6.1). It was suggested that properties such as those discussed (physical, historical, functional, legal and access) could be used to reclassify features and also to automatically generate some new features by aggregation for a variety of applications and end-uses.

8.4 Context-dependent pedestrian access

Access was identified as an important aspect of space in the built-environment, designed to host human activity (section 2.9.3). Applications of pedestrian access inside and outside buildings were described in sections 2.9.3.3 and 2.9.4. In particular, the discussion of emergency planning applications in section 2.9.4 illustrated that context-dependent pedestrian access was important (e.g. Pu and Zlatanova, 2005).

Section 4.7 reviewed approaches to handling access. It was proposed that the incorporation of a rich context-dependent model of access into a feature-based model of the built environment is enormously valuable. This is not only to support applications of pedestrian modelling and evacuation (common applications requiring a model of the built environment and pedestrian access), but also to support analyses of the relationships between accessibility and the combination of geometry, features,

activity and function. Analyses of the latter nature are likely to some interesting insights about built environments.

8.5 An integrated approach to storing geometry, features and access at the highest level of detail available

As implied by the previous sections, there is great strength in incorporating many facets of the environment into one description, because many of the applications reviewed would benefit from information about these various facets (section for example section 4.6.1). In addition, it is likely that the analysis of the relationships between these properties are likely yield important insights into the functioning of the human activity and the environment. Such analyses are not currently possible because of the lack of a unifying framework which can describe these properties at a regional or national scale.

8.6 Best output possible produced from incomplete data

In section 2.6.3, comparisons between the application domains of CAD and GIS were made. One of the differences identified was that CAD software tends to work with 'as designed' rather than 'as exists' structures; as such, CAD software has an emphasis on representing the precise geometry unlike GIS software. It is entirely possible to precisely survey the distinct boundaries of the real-world, the approach taken by surveyors. However, the requirement of having to perform a fully-resolved 3D survey for national mapping, is likely to a barrier to the production of a 3D mapping product.

This thesis proposed that existing vector mapping should used as a basis, and it should be annotated with selected height and surface morphological information. As data become available, they are added in an incremental fashion. The (2D) *spatial resolution is increased* as data become available and are added; for example, incorporating the interiors of some buildings. The *height model (3D geometry) is refined* by incorporating further height and surface morphological information as it becomes available; better 3D geometrical output can then be 'reasoned' by the '3DReasoner'. Thus, the framework manages incompleteness and the uncertainty associated with the incompleteness. Tools can then be designed to quantify the uncertainty, with these areas being flagged as requiring more data. The topological structuring of the model into layers compensates for some of the potential lack of heights, as explained in the next section. These topological constraints are essential for generating the 3D geometry.

8.7 A novel design for the multilayered 2.5D storage of these data

A 2.5D solution was proposed which allows existing 2D vector data to be exploited. Instead of there being one 'cartographic surface' upon which the various features are placed, there are multiple layers (patches), none of which have any priority. These layers (patches) topologically connect by access routes and their surface geometries are dynamically generated from the height information amongst the geometrical primitives in the database. Spot heights constrain the surface to an absolute height above the datum, relative heights constrain the surface relative to a point on the same or another surface, and morphological information includes vertical offsets in the surface and breaks of slope. There are rules which apply where data are under-resolved, based on observations from the built environment (the predominance of horizontal surfaces). This design allows for complicated relationships between features and the terrain; e.g. buildings with access points on different levels, bridges and underground and buildings on raised platforms.

These layers contain geometrical primitives which contain no real-world 'meaning' and which do not overlap within their layers. Features are modelled separately. A feature has a geometrical footprint, a set of attributes, a 3D extent described through parameters and may affect access. The geometrical footprint is described as a set of geometrical primitives which are time-stamped allowing any change in the footprint to be recorded over time. If a feature has a structural extent, is it parameterised. For example, a 'wall' feature has an attribute describing its heights (for its 3D extent) and information about how effective it is at being a barrier to access.

Access information is embedded into features. The pedestrian access resolver uses access and geometrical information embedded within the features which lie in the way, the geometry of the terrain through which the route passes, characteristics of the pedestrian making the journey, information about time embedded in features and information about the time at which access is attempted, to delineate an accessible area. It is then possible to generate geometrical networks from these.

8.8 Implications and future work

Chapter 7 demonstrated and evaluated the representational ability of the proposed model, which was found to perform well. This was particularly the case for the representation of the geometrical interaction between building features and a sloping terrain (case study 8; section 7.2.8).

Suggestions for further work are outlined below.

8.8.1 Implementation

Section 7.3.5 discussed problems with the practical use of the *specific implementation.* The process of inputting of data is awkward and the '3D reasoning' process is too slow to be of practical use without the long-term caching of the output (in the implementation, results are cached in RAM, which are lost when the application stops running). Although the requirement for dynamically interpolating and interpreting data from a smaller set when required does not help, this is an implementational problem which can be solved independently of the conceptual model.

8.8.1.1 Efficiency

As strongly hinted in the previous chapter, there is plenty of scope for implementing the system, such that computation times are more appropriate for its practical use. This is a necessity if the system is to be used for real-world applications, instead of the small proof-of-concept examples presented in this thesis. Some of empirical work suggested (such as assessing the relationship between access and function in the built-environment and investigating the scope for making inferences about the built-environment at different scales) require an efficient implementation of the framework to be populated with data over large geographical areas. This is likely to require a certain amount of redundancy in data storage, e.g. caching some of the derived information over long periods of time. Section 7.3.5.1 noted that under-resolved height data often led to long 3D reasoning computation times. In order to reduce this problem, rules could be imposed for specifying minimum amounts of data to improve the performance of the model. A quantitative analysis of the effect of under-resolved heights on computation time could be used to identify these rules.

8.8.1.2 Data entry

The data entry as implemented is clumsy and difficult to use. A more streamlined system is needed for the system to be used for real-world applications. This should be coupled with the development of data validation and interactive feedback tools. In particular, a validation procedure should take place to test whether the input data complies with data entry rules and also to test whether the topological linking of newly input geometry to other newly input geometry or geometry which already exists in the database, is valid. Also, interactive tools for identifying height conflicts and under-resolved areas are required.

8.8.2 Geometry

8.8.2.1 Time and geometry

Section 5.3.5 stated the assumption that pure geometry in the model does not change through time; it simply gets more refined. However, as section 7.3.4 showed, this is only true in a purely 2D sense. The history of variations in the surface morphology of the layers cannot be recorded. In order achieve this, the model would need be extended so that heights and morphological information were time-stamped in the same way as features.

- 'Node' entities need a history of height value change. Since it is possible that a node will have different roles over time, there must also be a history of changes between the various other entities in the hierarchy ('AbsHeightNode' and 'RelHeightNode'; figure 5.1.
- Since it is possible that a line will have different roles over time, 'Line' entities need a history of transformations between 'OffsetLine' and 'OffsetRelHeight-Lines and the heights and reference geometries of these (figure 5.1).
- The history of transformations of 'Polygon' entities to 'PolygonRamp' and 'PolygonStairs' (figure 5.1), because these have a bearing on the surface morphology. In addition, in order to allows holes to come into existence and go out of existence, polygons will need to have a history of existence, in the same way that features do.

An alternative solution would be to implement a full 3D geometrical model as discussed below.

8.8.2.2 3D geometry primitive model

A move to a 3D geometrical model in which the whole of 3D space is exhaustively decomposed into volumes rather than layers of polygons but retaining the other concepts of the conceptual model is a possibility for further work (the work in tetrahedral networks of Pilouk, 1996, may provide an appropriate model).

However, here, the concept of 'layers' is important as this represents the ground surface upon which human movement and activity takes place. The main conflict with the design principles for using a full 3D geometrical model is that x, y and z will be to be as fully-resolved as each other, whereas the 2.5D approach proposed, is suitable because only x and y must be fully resolved (within captured detail).

8.8.2.3 Functional surface geometrical constraints

In the conceptual model and prototype, the majority of the height and surface morphological information has come from the geometrical model i.e. from the 'Geometry' entity hierarchy. Subsidiary information has come from 'Door/Windows'/'Portals'

and 'Lifts'/'Teleports' (section 6.11.3.4). This additional minor source of information could be described as being functional constraints, because they are dependent on the function of a real-world conceptualisation of a feature.

The existing method of building the 3D geometry of stairs and ramps (section 6.11.3.3) can also been seen as being dependent on function. They both act as routes; they are built so that they are perpendicular to the direction of travel. For the direction of travel to be known, the entry and exit points are required (this is done through the use of breaklines, figure 6.57). This could be broadened to a wider range of features, for example:

- 'Kerbs' – height constrained to a height range (e.g. 1-15cm).
- 'Paved areas' – gradient not to exceed a certain gradient.
- 'Roads' and 'paths' – horizontal perpendicular to the path of travel.

The latter was attempted in the implementation. Using the TIN-based techniques described in section 7.5.2.1, the centreline was found and at regularly spaced intervals along the centreline, the existing terrain height was taken and added to the terrain point at both edges of the road perpendicular to the point along the centreline. The problems highlighted in the section about generating the centrelines dogged the experiment and it was not developed further.

8.8.2.4 Boundary uncertainty

As discussed in section 2.4, there is often uncertainty in the placement of boundaries. Some boundaries, particularly in the natural environment, have an inherent uncertainty (e.g. for lakes, this is dependent on factors such as climate, weather and local drainage patterns). However, in the tradition of surveying and GIS usually assumed that 2D boundaries are distinct.

The solution proposed relies on distinct boundary placement in 2D. Changes in 2D boundaries are handled, but not their uncertainty or fuzziness. The treatment of height and surface morphology is more fuzzy; uncertainty in height is managed and interpolated using height constraints, surface morphological information and layer topology.

It was also noted that often the presence of a feature and its topological relationships are more important than precise geometry. A system for storing and handling such topological relations might be appropriate. For example, if a feature is further north than another, but its position is not known, a 'north of' relation would be a more representationally correct than an inferred position.

8.8.3 Features

The conceptual model has a very 'stripped-down' approach to dealing with features. Features are simply prescribed by the data provider and classified as a particular

feature type. There is scope for future work on automated methods for classifying features and for analysing their change through time. Features can be classified based on their geometrical and attribute properties using principles described in 4.6.4.

The implementation can be modified with relative ease to allow all attributes to be timestamped. More fundamentally, managing the identities of features along a timeline is a difficult problem. For example, whether the identity of a building changes on its demolition and subsequent rebuilding while keeping the same address and function.

8.8.4 Access

The thesis developed principles and an approach to describing microscale pedestrian access. It is acknowledged that the precise factors (attributes) are illustrative rather than comprehensive. It is also acknowledged that models need to be fit for use for a specific set of applications and end-uses. Thus, an empirical analysis needs to be performed of access issues in buildings, in order to identify and validate a comprehensive set of properties affecting access.

Currently, the algorithms which delineate access are static in that they are run for a specific snapshot of time. However, during the time in which a pedestrian is moving, time progresses and accessible space changes. There may also be feedbacks and asymmetries, for example after leaving a building, a pedestrian may not be allowed back inside.

8.8.5 Applications

Applications which are in scope were identified in chapter 2 and it is expected that these will be supported by an efficient implementation of this framework. As stated throughout this thesis, many of the specific attributes chosen are application dependent; for example, the description of pedestrian access. Thus, in order to apply the model to specific applications, some empirical research needs to be carried out. For example, for describing pedestrian access in a built environment, a study of the nature of pedestrian access within the built environment should be used to validate and, if necessary, refine the details of the description of the state of pedestrian access.

There is great potential for developing both theory and practical applications for the framework, which can be used as a test-bed for novel research into the geometrical, topological and feature-based properties of built environments, in the wider context of geographical information science.

References

Al-Taha, K. (2001). Why time matters in cadastral systems. In A. Frank, J. Raper, and J. Cheylan (Eds.), *Life and motion of socio-economic units*, pp. 245–260. London: Taylor & Francis.

Allen, J. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM 26*(11), 832 – 843.

AutoDesk (2003). Autodesk Revit for retail. AutoDesk. [http://images.autodesk.com/apac_korea_main/files/3455842_RevitRetail_DP12_final.pdf].

Barr, R. (2004). OS MasterMap the vision: How Ordnance Survey led the world in the creation of a new type of national geospatial dataset. *GI News March/April*. [http://www.ginews.co.uk/0304_28.pdf].

Batty, M., D. Chapman, S. Evans, M. Hackley, S. Kueppers, N. Shoide, A. Smith, and P. Torrens (2001). Visualizing the city: Communicating urban design to planners and decision makers. In R. Brail and R. Klosterman (Eds.), *Planning Support Systems*, pp. 405–443. Redlands, California, USA: ESRI Press.

Batty, M. and S. Rana (2002). Reformulating space syntax: The automatic definition and generation of axial lines and axial maps. CASA Working Paper, Centre for Advanced Spatial Analysis (CASA), University College London.

Baumgart, B. (1972). Winged edge polyhedron representation. Stanford University.

Bell, D. (2005). *Software Engineering for Students: A programming approach* (4 ed.). Harlow, UK: Addison-Wesley.

Bennett, B. (2003). Foundations for an ontology of built environments. School of Computing, University of Leeds.

Bennett, B. (2005). Modes of concept definition and varieties of vagueness. *Applied Ontology 1*(1).

Billen, R. and S. Zlatanova (2001). 3D spatial relationships model: a useful concept to 3D cadastre? In P. van Oosterom, J. Stoter, and E. Fendel (Eds.), *Proceedings of the International Workshop on "3D Cadastres"*, Delft, The Netherlands, pp. 223–243. FIG.

Bittner, T., M. Donnelly, and S. Winter (2006). Ontology and semantic interoperability. In S. Zlatanova and D. Prosperi (Eds.), *Large-scale 3D data integration*, pp. 139–160. Boca Raton, FL, USA: CRC Press.

Bogaerts, T. and J. Zevenbergen (2002). Cadastral systems: alternatives. *Computers, Environment and Urban Systems 25*(4-5), 325–337.

Borland (1998). dBASE .DBF file structure. Borland. [http://community.borland.com/article/0,1410,15838,00.html].

Braeutigam, F., G. Müller, P. Nyfelt, and L. Mekenkamp (2003). Ozone users' guide. SMB GmbH. [http://sourceforge.net/projects/ozone/].

Braun, J. and M. Sambridge (1997). Modelling landscape evolution on geological time scales: a new method based on irregular spatial discretization. *Basin Research 9*(1), 27–52.

Breunig, M. and S. Zlatanova (2006). 3D geo-DBMS. In *Large-scale 3D data integration*, pp. 87–115. Boca Raton, FL, USA: CRC Press.

British Standards Institute (2000a). BS 7666-1:2000 spatial datasets for geographical referencing – part 1: Specification for a street gazetteer. British Standards Institute.

British Standards Institute (2000b). BS 7666-1:2000 spatial datasets for geographical referencing – part 2: Specification for a land and property gazetteer. British Standards Institute.

British Standards Institute (2000c). BS 7666-1:2000 spatial datasets for geographical referencing – part 3: Specification for addresses. British Standards Institute.

British Standards Institute (2000d). BS 7666-1:2000 spatial datasets for geographical referencing – part 4: Specification for recording public rights of way. British Standards Institute.

Brown, F., P. Rickaby, H. Bruhns, and P. Steadman (2000). Surveys of nondomestic buildings in four English towns. *Environment and Planning B: Planning and Design 27*, 11–24.

Bruhns, H. R., P. Steadman, H. Herring, S. Moss, and P. A. Rickaby (2000). Types, numbers, and floor areas of nondomestic premises in England and Wales, classified by activity. *Environment and Planning B 27*(5), 641–666.

Burrough, P. A. (1994). Natural objects with indeterminate boundaries. In P. A. Burrough and A. U. Frank (Eds.), *Concepts and languages in GIS; Geographic objects with indeterminate boundaries Frank*, Baden, Austria, pp. 3–28. London.

Campari, I. (1994). Uncertain boundaries in urban space. In P. A. Burrough and A. U. Frank (Eds.), *Concepts and languages in GIS; Geographic objects with indeterminate boundaries Frank*, Baden, Austria, [date unconfirmed], pp. 57–70. London.

Carkenord, B. (2001). Why build a logical data model. Embarcadero Technologies. [http://database.ittoolbox.com/white-papers/why-build-a-logical-data-model-923].

Chang, D. (2002). Spatial choice and preference in multilevel movement networks. *Environment and Behavior 34*(5), 582–615.

Church, R. L. and J. Marston (2003). Measuring accessibility for people with a disability. *Geographical Analysis 35*(1), 83–96.

Clover, R. (2000). VERTS synthetic urban environment development process – end to end. In *Interservice/Industry Training, Simulation and Education Conference 2000 (I/ITSEC-2000)*. [http://www.simsysinc.com/i_itsec00.htm].

Codd, E. (1970). A relational model for large shared data banks. *Communications of the ACM 13*(6), 377–387. [http://www.acm.org/classics/nov95/toc.html].

Cohn, A. (1995). A hierarchical representation of qualitative shape based on connection and convexity. In *Conference On Spatial Information Theory (COSIT)*, Semmering, Austria, pp. 311–326.

Cowen, D. (1990). GIS verus CAD verses DBMS: what are the differences? In D. Peuquet and D. Marble (Eds.), *Introductory Readings in Geographic Information Science*. London & New York.

Cullen, M. (2005). What have we started?! The geospatial future of buried services. *Civil Engineering Surveyor*. [http://www.dnf.org/Publications/Articles/Civil_Engineering_Surveyor2.pdf].

Cumberbatch, S. (2005). What lies beneath. *Surveyor*. [http://www.dnf.org/Publications/Articles/SVY0505p14-151.pdf].

Eastman, C. (1999). *Building product models: computer environments, supporting design and construction*. CRC Press. Boca Raton, FL, USA.

Egenhofer, M. and J. Herring (1994). Categorizing binary topological relations between regions, lines and points in geographic databases. In M. Egenhofer, D. Mark, and J. Herring (Eds.), *The 9-Intersection: Formalism and Its Use for Natural-Language Spatial Predicates*. National Center for Geographic Information and Analysis (NCGIA), University of California.

Egenhofer, M., J. Sharma, and D. Mark (1993). A critical comparison of the 4-intersection and 9-intersection models for spatial relations: formal analysis. In

Auto-Carto 11, Minneapolis; MN, pp. 1–11. American Society for Photogrammetry and Remote Sensing.

Ellul, C. and M. Haklay (2005). Beyond 9-intersections – modifications to a boundary-representation topological structure to identify additional relationships in 3D. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXVI*(2/W29).

ESRI (1998). ArcView Shapefile technical description. Environmental Systems Research Institute, inc. [http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf].

ESRI (2002). GIS and CAD—the right tool for the job. ESRI. [http://www.esri.com/library/whitepapers/pdfs/gis_and_cad.pdf].

ESRI (2003). ArcCAD user's guide. Environmental Systems Research Institute, inc. [http://support.esri.com/index.cfm?fa=knowledgebase.whitepapers.viewPaper&PID=39&MetaID=451].

Flanagan, D. (1996). *Java in a Nutshell* (3rd ed.). O'Reilly.

Fowler, R. and J. Little (1979). Automatic extraction of irregular network digital terrain models. *Proceedings of the 6th annual conference on Computer graphics and interactive techniques 13*(2), 199 – 207. [http://portal.acm.org/ft_gateway.cfm?id=807444&type=pdf&coll=GUIDE&dl=GUIDE&CFID=59138314&CFTOKEN=16141899].

Frank, A. (1994a). Qualitative topological relations and indeterminate boundaries. In P. A. Burrough and A. U. Frank (Eds.), *Concepts and languages in GIS; Geographic objects with indeterminate boundaries Frank*, Baden, Austria, pp. 153–154. London.

Frank, A. (2001). Socio-ecomonic units: their life and motion. In A. Frank, J. Raper, and J. Cheylan (Eds.), *Life and motion of socio-economic units*, pp. 21–34. London: Taylor & Francis.

Frank, A., J. Raper, and J. Cheylan (Eds.) (2001). *Life and motion of socio-economic units.* Geodata 8. London: Taylor & Francis.

Frank, A. U. (1994b). The prevalence of objects with sharp boundaries in GIS. In P. A. Burrough and A. U. Frank (Eds.), *Concepts and languages in GIS; Geographic objects with indeterminate boundaries Frank*, Baden, Austria, pp. 29–40. London.

Funtowicz, S. and J. Ravetz (1990). *Uncertainty and Quality in Science for Policy.* Dordrecht.: Kluwer Academic Publishers.

Goodchild, M. (2006). GIS and disasters: planning for catastrophe. *Computers Environment and Urban Systems 30*(3), 227–229.

Grün, A. (1997). Automation in building reconstruction. *Photogrammetrische Woche, University of Stuttgart.*

Guibas, L. and J. Stolfi (1985). Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics 4*(2), 74–123.

Gwynne, S., E. Galea, M. Owen, P. Lawrence, and L. Filippidis (1999). A review of the methodologies used in the computer simulation of evacuation from the built environment. *Building and Environment 34*(6), 741–749.

Hansen, H. (2001). A quasi-four dimensional database for the built environment. *Lecture Notes in Computer Science ISSU*(2181), 48–59.

Hillier, B. and J. Hanson (1984). *The social logic of space.* Cambridge, UK: Cambridge University Press.

HMLR (2004). Land Registry public guide 6: boundary questions. HMLR. [http://www.landregistry.gov.uk/assets/library/documents/public_guide_006.pdf].

Hoffmann, C. (1989). *Geometric & Solid Modeling.* California: Morgan Kaufmann Publishers Inc.

Holtier, S., J. Steadman, and M. Smith (2000). Three-dimensional representation of urban built form in a GIS. *Environment and Planning B: Planning and Design 27*, 51–72.

Hornsby, K. and M. J. Egenhofer (1997). Qualitative representation of change. *Lecture Notes in Computer Science*, 15–34.

Howard, R., D. Wager, and E. Winterkorn (1994). Guidance on selecting selecting energy programs. Construction Industry Computing Association.

Hudson-Smith, A. and S. Evans (2003). Virtual cities: from CAD to 3D-GIS. In P. Longley and M. Batty (Eds.), *Advanced Spatial Analysis - The CASA Book of GIS*, pp. 41–60. ESRI Press.

Hwang, J. and K. Koile (2005). Heuristic Nolli map: a preliminary study in representing the public domain in urban space. In S. Batty (Ed.), *Proceedings of Computers in Urban Planning and Urban Management (CUPUM)*, London, UK. [http://128.40.59.163/cupum/searchPapers/papers/paper273.pdf].

I&DeA (2004, April 2004). Project Acacia: multi-way cross-referencing. Improvement & Development Agency. [http://www.cofrestrfatir.gov.uk/assets/library/documents/acacia_nlpg_submission.pdf].

ISO (1994). Industrial automation systems and integration. Product data representation and exchange. Description methods: the EXPRESS language reference manual. International Standards Organisation.

Jones, N., S. Wright, and D. Maidment (1990). Watershed delineation with triangle-based terrain models. *Journal of Hydraulic Engineering 16*(10), 1232–1251.

Julstad, B. and A. Ericsson (2001). Property formation and three-dimensional property units in Sweden. In P. van Oosterom, J. Stoter, and E. Fendel (Eds.), *International workshop on 3D cadastres*, Delft, The Netherlands, pp. 173–189. International Federation of Surveyors, Denmark.

Khemlani, L. (2002). Comparing Revit and Autodesk Architectural Desktop: Part 2. *CADENCE AEC Tech News 73.* [http://cadence.advanstar.com/newsletter/aec/0402_2.html].

Khemlani, L. (2003a). Building information modeling gains momentum. *CADENCE AEC Tech News 90.* [http://cadence.advanstar.com/newsletter/aec/0103_2.html].

Khemlani, L. (2003b). A "federated" approach to building information modeling. *CADENCE AEC Tech News 94.* [http://cadence.advanstar.com/newsletter/aec/0303_2.html].

Khemlani, L. (2004). The IFC building model: A look under the hood. AECbytes Feature

Kirkby, S., S. Pollitt, and P. Eklund (1996). Implementing shortest-path algorithm in a 3D GIS environment. In M. Kraak and M. Molenaar (Eds.), *Advances in GIS Research II*, pp. 437–448. Delft, The Netherlands,.

Kolbe, T., G. Gröger, and L. Plümer (2005). CityGML interoperable access to 3D city models. In P. van Oosterom, S. Zlatanova, and E. Fendel (Eds.), *Proceedings of the Int. Symposium on Geo-information for Disaster Management*, Delft, The Netherlands. Springer Verlag. [http://www.citygml.org/docs/Gi4Dm_2005_Kolbe_Groeger.pdf].

Kottman, C. (1999, 23 March, 1999). Topic 5 - features. Open Geospatial Consortium. [http://portal.opengeospatial.org/files/index.php?artifact_id=890].

Kraak, M. (2003). The space-time cube revisited from a geovisualization perspective. In *Proceedings of the 21st International Cartographic Conference (ICC)*, Durban, South Africa, pp. 1988–1995. [http://www.itc.nl/library/Papers_2003/art_proc/kraak.pdf].

Krüger (1979). An approach to built-form connectivity at an urban scale: system description and its representation. *Environment and Planning B 6*(1), 67–88.

Kwan, M. P. (2002). Time, information technologies, and the geographies of everyday life. *Urban Geography 23*(5), 471–482. [http://geog-www.sbs.ohio-state.edu/faculty/mkwan/Paper/Kwan_UG2002.pdf].

Kwan, M. P. and J. Lee (2005). Emergency response after 9/11: the potential of real-time 3D gis for quick emergency response in micro-spatial environments. *Computers Environment and Urban Systems 29*(2), 93–113.

Langran, G. (1992). *Time in Geographical Information Systems.* London, UK: Taylor & Francis.

Lawrence, P. (1989). Translating anthropological concepts into architectural practice. In S. Low and E. Chambers (Eds.), *Housing, culture and design*, pp. 89–114. Philadelphia, USA: University of Pennsylvania Press.

Lee, J. (1991). Comparison of existing methods for building triangular irregular network models of terrain from grid digital elevation models. *International Journal of Geographical Information Systems 5*, 267–285.

Lee, J. (2004a). 3D GIS for geo-coding human activity in micro-scale urban environments. *Lecture Notes in Computer Science* (3234), 162–178.

Lee, J. (2004b). A spatial access-oriented implementation of a 3-D GIS topological data model for urban entities. *Geoinformatica 8*(3), 237–264.

Lee, J. and M. Kwan (2005). A combinatorial data model for representing topological relations among 3D geographical features in micro-spatial environments. *International Journal of Geographical Information Science 19*, 1039–1056.

Lee, Y. (1990). Geographic information systems for urban applications: problems and solutions. *Environment and Planning B 17*(4), 463–473. [http://www.envplan.com/epb/abstracts/b17/b170463.html].

Lewis, R. (1996). *Generating three-dimensional building models from two-dimensional architectural plans.* Masters, University of California.

Lewis, R. and C. Séquin (1998). Generation of 3D building models from 2D architectural plans. *Computer Aided Design 30*(10), 765–780.

Liu, S. and X. Zhu (2004). Accessibility analyst: an integrated GIS tool for accessibility analysis in urban transportation planning. *Environment and Planning B 31*(1), 105–124.

Lynch, K. (1960). *The Image of the City.* Cambridge, Massachusetts, USA: Havard University Press.

Maksimchuk, R. and E. Naiburg (2003). Entity relationship modeling with UML. *DM Direct Newsletter*. [http://www.dmreview.com/article_sub.cfm?articleId=6268].

Mäntlyä, M. (1988). *An Introduction to Solid Modeling.* Maryland, USA: Computer Science Press.

Marshall, S. (2005). *Street patterns.* Abbington, Oxon, UK: Spon Press.

Maynard, J. (1989). Great oaks from little acorns. *Land and Minerals Surveying 7*, 584–589. [http://www.boundary-problems.co.uk/frameoslr100.htm].

Maynard, J. (2001). Digital boundaries in England and Wales. *Surveying World 9*(4). [http://www.boundary-problems.co.uk/maindigbdies.htm].

Meijers, M., S. Zlatanova, and N. Pfeifer (2005). 3D geo-information indoors: Structuring for evacuation. In *Proceedings of Next Generation 3D City Models*, Bonn, Germany. [http://www.gdmc.nl/publications/2005/3D_indoor_geoinformation.pdf].

Miller, R. (2004). Practical UML: A hands-on introduction for developers. [http://bdn.borland.com/article/0,1410,31863,00.html].

Molenaar, M. (1998). *An Introduction to the Theory of Spatial Object Modelling.* Research Monographs in GIS Series. London: Taylor & Francis.

Morrison, J. (1994). The paradigm shift in cartography: The use of electronic technology, digital spatial and future needs. In T. Waugh and R. Healey (Eds.), *Advances in GIS research, 6th International Symposium on Spatial Data Handling*, Edinburgh, UK, pp. 1–15. Taylor & Francis.

Morrison, J. (1997). Topographic mapping in the twenty-first century. In D. Rhind (Ed.), *Framework for the World*, pp. 14–27. Cambridge, UK: Geoinformation Internation, Pearson Professional Ltd.

Murray, K. (2002). A new geo-information framework for Great Britain. In *FIG XXII International Congress*, Washington, D.C. USA,. [http://www.fig.net/pub/fig_2002/Ts3-3/TS3_3_murray.pdf].

Newell, R. and T. Sancha (1990). The difference between CAD and GIS. Smallworld. [http://emea.smallworld.co.uk/support/techpaper/tp2.html].

Okunuki, K., R. Church, and J. Marston (1999). A study on a system for guiding of the optimal route with a hybrid network and grid data structure. In *Papers and Proceedings of the Geographic Information Systems Association*, Volume 8, Japan, pp. 135–138.

Ordnance Survey (1996). TOPO-96 capture and maintenance. Ordnance Survey.

Ordnance Survey (2001). OS MasterMap real-world object catalogue. Ordnance Survey. [http://www.ordnancesurvey.co.uk/products/osmastermap/faqs/Docs/realWorldObjectCatalogue.pdf].

Ordnance Survey (2002). Land-Line user guide. Ordnance Survey. [http://www.ordnancesurvey.co.uk/products/landline/pdf/lluserguide.pdf].

Ordnance Survey (2004). The Digital National Framework - evolving a framework for interoperability across all kinds of information. Ordnance Survey. [http://www.ordnancesurvey.co.uk/oswebsite/aboutus/reports/dnf_white_paper.pdf].

Ordnance Survey (2005). OS MasterMap user guide, part 2 (reference section). Ordnance Survey. [http://www.ordnancesurvey.co.uk/products/osmastermap/userguides/docs/userguidepart2.pdf].

Paay, J. and J. Kjeldskov (2005). Understanding and modelling built environments for mobile guide interface design. *Behaviour & Information Technology 24*(1), 21–35.

Papamichael, K. (1997). Designers and information overload: a new approach. *Advanced Buildings Newsletter 1*(18).

Papamichael, K., J. La Porta, and H. Chauvet (1997). Building Design Advisor: automated integration of multiple simulation tools. *Automation in Construction 6*(4), 341–352.

Pilouk, M. (1996). *Integrated modelling for GIS.* PhD. thesis, International Institute for Geo-Information Science and Earth Observation (ITC).

Pluijmers, Y. (2002). The economic impacts of open access policies for public sector spatial information. In *FIG XXII International Congress*, Washington, D.C. USA. [http://www.fig.net/pub/fig_2002/Ts3-6/TS3_6_pluijmers.pdf].

Pu, S. and S. Zlatanova (2005). Evacuation route calculation of inner buildings. In P. van Oosterom, S. Zlatanova, and E. M. Fendel (Eds.), *International symposium on geo-information for disaster management*, Delft, The Netherlands, pp. 1143–1162. Springer.

Pun-Cheng, L. and Y. Lee (2004). An addressing model for three-dimensional city properties in Hong Kong. *URISA Journal 16*(1), 39–45. [http://urisa.org/Journal/protect/Vol16No1/Pun-Cheng.pdf].

Ratti, C. (2002). *Analysis for Environmental Prediction.* Ph.d. thesis, University of Cambridge.

Regnauld, N. and W. Mackaness (2006). Creating a hydrographic network from its cartographic representation: a case study using ordnance survey mastermap data. *International Journal of Geographical Information Science 20*(6), 611 – 631.

Rhind, D. (1997). *Framework for the World.* Cambridge, UK: Geoinformation Internation, Pearson Professional Ltd.

Rousseaux, F. (2003). Enrichment of a DTM with 3D vector data. In J. Wood (Ed.), *Proceedings of GISRUK 2003*, London, UK, pp. 82–85.

Sakkas, N. and J. Pérez (2005). Elaborating metrics for the accessibility of buildings. *Computers, Environment and Urban Systems.*

Sellis, T. (1999). Research issues in spatio-temporal database systems. In H. Guting, D. Papadias, and F. H. Lochovsky (Eds.), *Spatial databases*, Hong Kong; China, pp. 5–11. Berlin.

Shewchuk, J. (1996). Triangle: Engineering a 2D quality mesh generator and delaunay triangulator. *Lecture Notes in Computer Science 1148*, 203–222. [http://www.cs.cmu.edu/~quake/tripaper/triangle0.html].

Shoshani, U., M. Benhamu, E. Goshen, S. Denekamp, and R. Bar (2005). A multi layers 3D cadastre in Israel: a research and development project: recomendations. In *3D Cadastre: FIG Working Week 2005 and GSDI-8*, Cairo, Egypt. [http://www.mapi.gov.il/files/FIG2005.pdf].

Slingsby, A. (2002). *An object-orientated approach to hydrological modelling based on a triangular irregular network.* Msc thesis (unpublished), The University of Edinburgh.

Slingsby, A. (2003). An object-orientated approach to hydrological modelling using triangular irregular networks. In J. Wood (Ed.), *GISRUK*, London, UK, pp. 92–98.

Slingsby, A. (2005). Pedestrian accessibility in the built environment in the context of feature-based digital mapping. In S. Batty (Ed.), *Proceedings of Computers in Urban Planning and Urban Management (CUPUM)*, London, UK. [http://128.40.59.163/cupum/searchPapers/papers/paper305.pdf].

Slingsby, A. and P. Longley (2006). A conceptual framework for describing microscale pedestrian access in the built environment. In G. Priestnall and P. Aplin (Eds.), *GISRUK*, Nottingham, UK, pp. 166–173.

Slingsby, A., P. Longley, and C. Parker (2004a). Aspects of the design of a three-dimensional national mapping data framework. In A. Lovett (Ed.), *GISRUK*, Norwich, UK, pp. 241–244.

Slingsby, A., P. Longley, and C. Parker (2004b). A new framework for feature-based digital mapping in three dimensional space. In A. Lovett (Ed.), *Innovations in GIS 12: GIS for Environmental Decision Making.* CRC Press.

Sloan, S. (1987). A fast algorithm for constructing delaunay triangulations in the plane. *Advances in Engineering Software 9*(1), 34–55.

Steadman, J. (1983). *Architectural Morphology.* London: Pion Limited.

Steadman, P., H. Bruhns, S. Holtier, B. Gakovic, P. Rickaby, and F. Brown (2000). A classification of built forms. *Environment and Planning B: Planning and Design 27*, 73–91.

Steadman, P., H. Bruhns, and P. Rickaby (2000). An introduction to the national Non-Domestic Building Stock database. *Environment and Planning B: Planning and Design 27*, 3–10.

Steadman, P. and P. Rickaby (Eds.) (2000). *A database and model of energy use in the nondomestic building stock of England and Wales (reprinted from Environment and Planning B, volume 27)*. Pion.

Steed, A., E. Frecon, D. Pemberton, and G. Smith (1999). The London Travel Demonstrator. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pp. 50–57. ACM Press.

Stoter, J. (2002). From 2D parcels to 3D registrations - 3D cadastres: state-of-the-art. *GIM International*, 12–15.

Stoter, J. (2004). *3D cadastre*. Ph.d. [http://www.itc.nl/library/Papers_2004/phd/stoter.pdf].

Stoter, J., M. Salzmann, P. van Oosterom, and P. van der Molen (2002). Towards a 3D cadastre. In *FIG XXII International Congress*, Washington, D.C., USA.

Stoter, J. and P. van Oosterom (2006). *3D Cadastre in an International Context*. Boca Raton, FL, USA: CRC Press.

Stoter, J. E. and H. D. Ploeger (2003). Property in 3D-registration of multiple use of space: current practice in holland and the need for a 3D cadastre. *Computers Environment and Urban Systems 27*(6), 553–570.

Talen, E. (2002). Pedestrian access as a measure of urban quality. *Planning Practice & Research 17*(3), 257278. [http://taylorandfrancis.metapress.com/media/m3t1fan03jdqql4a2q7p/contributions/n/e/p/5/nep54l3xcyhn8p64.pdf].

Tao, C. (2006). 3D data acquisition and object reconstruction for AEC/CAD. In S. Zlatanova and D. Prosperi (Eds.), *Large-scale 3D data integration*, pp. 39–56. Boca Raton, FL, USA: CRC Press.

Thompson, R. (2006). 3D topological framework for robust digital spatial models. In *Large-scale 3D data integration*, pp. 177–209. Boca Raton, FL, USA: CRC Press.

Tinkler, K. (1977). An introduction to graph theoretical methods in geography. *Concepts and Techniques in Modern Geography (CATMOG) 14*.

Tucker, G., S. Lancaster, N. Gasparini, R. Bras, and S. Rybarczyk (2001). An object-oriented framework for distributed hydrologic and geomorphic modeling using triangulated irregular networks. *Computers and Geosciences 27*(8), 959–973.

Urban, S. and S. Dietrich (2002). Using UML class diagrams for a comparative analysis of relational, object-oriented, and object-relational database mappings. In *Computer science education; SIGCSE 2003*, Reno, NV, pp. 21–25. Acm. [http://doi.acm.org/10.1145/611892.611923].

Valstad, T. (2001). The Oslo method: a practical approach to register 3D properties. In P. van Oosterom, J. Stoter, and E. Fendel (Eds.), *International workshop on 3D cadastres*, Delft, The Netherlands, pp. 1–8. International Federation of Surveyors, Denmark.

van der Molen, P. (2001). Institutional aspects of 3D cadastre. In P. van Oosterom, J. Stoter, and E. Fendel (Eds.), *International workshop on 3D cadastres*, Delft, The Netherlands, pp. 53–65. International Federation of Surveyors, Denmark.

van der Molen, P. (2003). Some experiences on the change of organisations for mapping and cadastre. [http://www.itc.nl/library/Papers_2003/non_peer_conf/vandermolen_some.pdf].

van Oosterom, P., J. Stoter, and E. Jansen (2006). Bridging the worlds of CAD and GIS. In *Large-scale 3D data integration*, pp. 9–36. Boca Raton, FL, USA: CRC Press.

van Oosterom, P., S. Zlatanova, and E. Fendel (Eds.) (2005). *Geo-information for disaster management.* Berlin, Germany: Springer-Verlag.

Wallace, J. and I. Williamson (2006). Developing cadastres to service complex property markets. *Computers Environment and Urban Systems*. [http://dx.doi.org/10.1016/j.compenvurbsys.2005.08.007(inpress)].

Wilson, P. (1998). STEP and EXPRESS. In *Workshop on Distributed Information, Computation and Process Management for Scientific and Engineering Environments*, Hyatt Dulles, Herndon, Virginia, USA. [http://deslab.mit.edu/DesignLab/dicpm/step.html].

Worboys, A. and M. Duckham (2004). *GIS: A computing perspective* (2 ed.). CRC Press.

Yildirim, V. and T. Yomralioglu (2004). An address-based geospatial application. In *FIG Working Week 2004*, Athens, Greece.

Zlatanova, S. (2000). *3D GIS for Urban Development.* PhD thesis, ITC publication 69, ISBN 90-6164-178-0. [http://www.gdmc.nl/zlatanova/PhDthesis/].

Zlatanova, S. and D. Prosperi (2006). *Large-scale 3D data integration.* Boca Raton, FL, USA: CRC Press.